Viability in State-Action Space

Connecting Morphology, Control, and Learning

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M.Sc. Steve Heim
aus Lugano / Schweiz

Tübingen
2019

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	20.02.2020
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Dr. Alexander Badri-Spröwitz
2. Berichterstatter:	Prof. Martin Giese

Abstract

How can we enable robots to learn control model-free and directly on hardware?

Machine learning is taking its place as a standard tool in the roboticist's arsenal. However, there are several open questions on how to learn control for physical systems. This thesis provides two answers to this motivating question.

The first is a formal means to quantify the inherent robustness of a given system design, prior to designing the controller or learning agent. This emphasizes the need to consider both the hardware and software design of a robot, which are inseparably intertwined in the system dynamics.

The second is the formalization of a safety-measure, which can be learned model-free. Intuitively, this measure indicates how easily a robot can avoid failure, and enables robots to explore unknown environments while avoiding failures.

The main contributions of this dissertation are based on viability theory. Viability theory provides a slightly unconventional view of dynamical systems: instead of focusing on a system's convergence properties towards equilibria, the focus is shifted towards sets of failure states and the system's ability to avoid these sets. This view is particularly well suited to studying learning control in robots, since stability in the sense of convergence can rarely be guaranteed during the learning process.

The notion of viability is formally extended to state-action space, with viable sets of state-action pairs. A measure defined over these sets allows a quantified evaluation of robustness valid for the family of all failure-avoiding control policies, and also paves the way for enabling safe model-free learning.

The thesis also includes two minor contributions. The first minor contribution

is an empirical demonstration of shaping by exclusively modifying the system dynamics. This demonstration highlights the importance of robustness to failures for learning control: not only can failures cause damage, but they typically do not provide useful gradient information for the learning process.

The second minor contribution is a study on the choice of state initializations. Counter to intuition and common practice, this study shows it can be more reliable to occasionally initialize the system from a state that is known to be uncontrollable.

Zusammenfassung

Wie können wir Robotern ermöglichen, modellfrei und direkt auf der Hardware zu lernen?

Das maschinelle Lernen nimmt als Standardwerkzeug im Arsenal des Robotikers seinen Platz ein. Es gibt jedoch einige offene Fragen, wie man die Kontrolle über physikalische Systeme lernen kann. Diese Arbeit gibt zwei Antworten auf diese motivierende Frage.

Das erste ist ein formales Mittel, um die inhärente Robustheit eines gegebenen Systemdesigns zu quantifizieren, bevor der Controller oder das Lernverfahren entworfen wird. Dies unterstreicht die Notwendigkeit, sowohl das Hard- als auch das Software-Design eines Roboters zu berücksichtigen, da beide Aspekte in der Systemdynamik untrennbar miteinander verbunden sind.

Die zweite ist die Formalisierung einer Sicherheitsmaß, die modellfrei erlernt werden kann. Intuitiv zeigt diese Maß an, wie leicht ein Roboter Fehlschläge vermeiden kann. Auf diese Weise können Roboter unbekannte Umgebungen erkunden und gleichzeitig Ausfälle vermeiden.

Die wichtigsten Beiträge dieser Dissertation basieren sich auf der Viabilitätstheorie. Viabilität bietet eine alternative Sichtweise auf dynamische Systeme: Anstatt sich auf die Konvergenzeigenschaften eines Systems in Richtung Gleichgewichte zu konzentrieren, wird der Fokus auf Menge von Fehlerzuständen und die Fähigkeit des Systems, diese zu vermeiden, verlagert. Diese Sichtweise eignet sich besonders gut für das Studium der Lernkontrolle an Robotern, da Stabilität im Sinne einer Konvergenz während des Lernprozesses selten gewährleistet werden kann.

Der Begriff der Viabilität wird formal auf den Zustand-Aktion-Raum erweitert, mit Viabilitätsmengen von Staat-Aktionspaaren. Eine über diese Mengen definierte Maß ermöglicht eine quantifizierte Bewertung der Robustheit, die für die Familie aller fehlervermeidenden Regler gilt, und ebnet den Weg für ein sicheres, modellfreies Lernen.

Die Arbeit beinhaltet auch zwei kleinere Beiträge. Der erste kleine Beitrag ist eine empirische Demonstration der Shaping durch ausschließliche Modifikation der Systemdynamik. Diese Demonstration verdeutlicht die Bedeutung der Robustheit gegenüber Fehlern für die Lernkontrolle: Ausfälle können nicht nur Schäden verursachen, sondern liefern in der Regel auch keine nützlichen Gradienteninformationen für den Lernprozess.

Der zweite kleine Beitrag ist eine Studie über die Wahl der Zustandsinitialisierungen. Entgegen der Intuition und der üblichen Praxis zeigt diese Studie, dass es zuverlässiger sein kann, das System gelegentlich aus einem Zustand zu initialisieren, der bekanntermaßen unkontrollierbar ist.

Acknowledgements

The road to graduation is long and laborious, but never lonely. It is the countless thought-provoking conversations I have had over the years that have made this choice in life worthwhile. Whether contemplating the woes of academia over a drink in the wee hours after a conference, discussing the latest results and planning next steps in the lab, or bouncing ideas back and forth over a ping-pong table, these conversations are what have led me to where I am, in research and in life. To each of you who have provoked my beliefs, stoked my imagination, tickled my thinking-box, thank you. I hope we meet again.

Contents

«Vous me demandez de vous prédire les phénomènes qui vont se produire.
Si, par malheur, je connaissais les lois de ces phénomènes, je ne pourrais
y arriver que par des calculs inextricables et je devrais renoncer à vous répondre;
mais, comme j'ai la chance de les ignorer, je vais vous répondre tout de suite.
Et, ce qu'il y a de plus extraordinaire, c'est que ma réponse sera juste.»

Science et Méthodes
Henri Poincaré

"To tell the truth, we don't do it because it is useful but because it is amusing."

Muscular Movement in Man
Archibald Vivian Hill

Chapter 1

Introduction

This cumulative thesis presents the four first-author papers I published during my Ph.D. studies in the Dynamic Locomotion Group of the Max Planck Institute for Intelligent Systems, Stuttgart [1]–[4]. The main contribution of these studies is an extension of viability theory into state-action space and a measure over viable sets in this space. The resulting mathematical objects open a path to new tools, including tools to rigorously quantify the robustness of a system prior to controller design [2] and model-free safe learning [3].

Each paper is attached in the appendix. The thesis is organized as follows: in Chapter 1 (this chapter), I introduce the overall motivation and objectives. Chapter 2 concisely covers background knowledge and related work. Chapter 3 summarizes each publication, my contributions, and suggests an order in which to read them. Finally, in Chapter 4, I discuss the relevance of the publications in relation to the overall theme and motivation and the open challenges they raise.

1.1 Overall Motivation

Dynamics are certainly one of the most fascinating domains to study, and motion is dynamics embodied. To observe motion, whether the graceful leap of a squirrel from one branch to another, or the gravity-defying feats of an aerobatics pilot, is a pleasure in and of itself. To understand motion, even more so.

If this were not motivation enough, motion and mobility are entrenched in every aspect of modern society. It is our ability to move people, goods, and information quickly, cheaply, and reliably that enables a global economy and society. The advent of mobile robots that can automate mobility promises to once again change the scales of what we consider far or 'just around the corner'. Automation will allow mobility at a lower cost and in situations where it is unsafe or otherwise undesirable to employ humans. Even without an increase in speed, automation can reduce the time involving the user to virtually nil. Nonetheless, for this to become an every-day reality requires a level of robustness and reliability that we can so far only obtain in controlled laboratory settings, where models can be refined to high accuracy.

Robustness to uncertainties is critical for deploying robots in the real world for many reasons. Since sensing is always noisy, a controller needs to be robust to noisy state estimation. Controllers are typically designed based on a model of the robot and the world. Since all models are wrong [5], some amount of robustness is always necessary, even in the most benign settings. Even with a very well chosen model and excellent system identification, the accuracy of the model used will decay over time. The robot's dynamics will change due to wear and tear, and the world the robot moves in will change. In realistic field deployments, it will become common for the robot to be used and even modified in unforeseen ways. Robustness will also help lower manufacturing costs since performance will be more tolerant of imprecise manufacturing. Finally, as will be discussed in more detail in this thesis, robustness plays a crucial role in enabling *learning* control.

Machine learning, especially reinforcement learning, is gaining a lot of traction in the community of robotics. Learning control directly from data can circumvent the inaccuracies of models, and can also allow the robot to continuously adapt to changes in the real world. A general learning algorithm could also reduce the engineering effort required to deploy a new robot, as the controller design could be automated. However, we still lack learning algorithms that generalize well, are data-efficient, and adequately consider practical issues such as safety.

A less studied aspect of the same problem is how the system itself should be

designed. Although there is some published evidence [6], [7] that appropriate system dynamics can significantly facilitate learning, as well as ample informal consensus[1], there have been few formal studies that rigorously show show how morphology, control, and learning are connected. This formalization is the main motivation of this thesis.

1.2 Objectives

'How can we enable robots to learn control model-free and directly on hardware?'

I believe robots of the future will be able to effectively leverage models, and therefore make extensive use of both simulation and optimal control. Nevertheless, I have chosen a guiding question that explicitly excludes these two tools for a very deliberate reason: it moves the focus to aspects that I believe have not received sufficient attention.

First and foremost is the necessity for robustness. Indeed, the absence of a model can be loosely thought of as having a model over which we have absolutely no certainty. Learning directly on hardware also emphasizes the need for robustness, as failures become a significant problem. Failures can cause damage to the robot or world, often require time-consuming resets, and perhaps most importantly, they typically do not provide rich gradient information.

Second, as anyone who has worked on physical robots can confirm, the hardware design and implementation immediately take a central role. Robots that are appropriately designed and well manufactured are a lot easier to work with and greatly alleviate a control designer's work[2]. When learning control, it is reasonable to expect a learning agent to reap the same benefits, even though we cannot

[1] At every conference I have attended that had a mixed audience, a biologist will point out to the engineers that animals often fall, and this is important for learning. The engineers then invariably agree that robots should be designed to allow for this, and that's the end of the story.

[2] I recently spoke with a researcher at Google Brain, where they have 5 Ghost Robotics Minitaur quadruped robots, some of which were produced in different batches. Due to the differences in production, they behave slightly differently, and the researcher confirmed they have their favorites for testing learning algorithms.

fully describe what these benefits are.

More specifically, we will aim to formalize an understanding of how morphology and system design influence how easy it is to design or learn a control policy. Furthermore, we will aim for a more principled understanding of how a learning agent can learn despite failures, and perhaps more importantly, directly reason about failure.

Chapter 2

Preliminaries

This chapter introduces relevant fields to help readers from potentially different backgrounds to quickly gain enough knowledge to understand the context, relevance, and importance of the contributions of each paper, in particular as related to dynamic legged locomotion. Covered are the basics of viability theory, legged locomotion, and reinforcement learning.

2.1 Viability theory

The core of my work is based on viability theory, first pioneered by Jean-Pierre Aubin [8]. The development of viability theory is largely motivated by observations of dynamical systems in nature and society, whose behavior somehow avoids chaos or pure randomness, yet never seem to settle at a resting state, an equilibrium. Some examples are Darwinian evolution, economics, or politics of the state. There is no equilibrium state for these systems, or at least, none that we can identify or foresee. The classical mathematical tools based on convergence to such an equilibrium state are therefore ill-suited to describe them, although they can be manipulated for the purpose: for example, by assuming a quickly time-varying equilibrium state, which the system chases but is never able to reach.

Viability theory provides a more appropriate and direct description. First, a set of failure states is defined, which the system must be able to avoid. We will

think of this failure set as a set of absorbing states. From this naturally emerges the *viability kernel*, the maximal set of states from which there exist control inputs that keep the system from entering the failure set for all time. In other words, if the system ever leaves the viability kernel, it can no longer return inside of it and is doomed to enter the failure set within finite time.

Compared to mathematical objects based on the notion of convergence, such as regions of attraction [9, (cf. 6.4)] or contractions [10], conditions for viability are weaker (they provide no convergence properties) but more general.

Two other related sets which depart from the notion of convergence are *back-reachable sets* [11] and *controllable sets* [12]. For backreachable sets, we start by defining a target set. The backreachable set is the set of states from which there exist control inputs that guide the system into the target set within finite time. Since being able to reach this target set directly implies being able to avoid the failure set, the back-reachable set is always a subset of the viability kernel. Controllable sets require that every point in the set is reachable from any other point of the set. In other words, the controllable set must be a backreachable set for all subsets of the controllable set itself. Controllability is a more general statement than backreachability in that it does not require the definition of a target state, but also a stronger statement: not all backreachable sets are controllable sets. By the same quality of backreachable sets, controllable sets are also subsets of the viability kernel.

Viability theory, therefore, allows us to make very general statements of a dynamical system's behavior, without making many assumptions. This generality comes at a price: computing viable sets is typically computationally expensive. For the types of dynamics we are interested in, we will resort to gridding and brute force. In this thesis, I will use the concepts of viability theory, but we will not concern ourselves with algorithms that scale to higher dimensions. For the interested reader, I recommend starting with [13] for a clear and practical application, [14] for an example of sample-driven approximations, and [11] for an overview of tools for the computation of the related backreachable sets, which can often be used in lieu of the viability kernel.

2.1.1 Why use viability for legged locomotion?

The study of legged locomotion has come a long way by thinking of gaits as stable limit-cycles: periodic orbits through state space, towards which nearby orbits converge. This view allows us to explain various observations with a well developed mathematical language: nonlinear dynamics and bifurcations [9].

For example, the remarkable stability of running dynamics [15] can be understood by studying basins of attraction of a specific gait [16], [17]. The choice of and transition between different gaits depending on locomotion speed or morphology can be understood via bifurcation studies [18]–[20].

However, these tools hinge on the existence of a limit-cycle, and therefore are ill-suited for understanding unsteady or erratic motion, which are also commonly observed [21], [22]. Even when it is reasonable to assume the existence of a limit-cycle, for example when walking or running on a treadmill at a steady speed, it is often difficult to fit recorded time-series data to these assumptions [23], [24]. To gain further insight into locomotion, we need tools that can reason about stability in the sense of avoiding falls instead of in the sense of convergence [25].

Viability theory provides a natural description for this type of stability. Furthermore, viability makes no assumptions on the control policy or the task at hand. This makes it appropriate for analyzing morphology separately from the control policy, as in [2].

In the field of legged robotics and control, there have also been developments which move away from the perspective of strict convergence. Byl and Tedrake [26] relax the requirement of convergence with the probabilistic concept of *metastability*, and compute the mean first-passage time for several simple models of walking. This metric takes into account the probability of failure from any sources of uncertainty, and can effectively serve as a metric of task-level robustness. However, the structure of how the system dynamics relate to failures is not exposed. Furthermore, any change in the uncertainty, such as the distribution of disturbances, requires the mean first-passage time to be recomputed from scratch.

Capture points and capture regions [27] build on the ideas of backreachability to find effective yet simple foot placements that can come to a standstill. In this

context, a *capture point* represents a foot placement that takes the system to a *captured state*: standing, static equilibrium. An *n-step capture region* is the set of foot placements that take the system to any state, from which the captured state can be reached within $n-1$ steps. The core idea is that it is not necessary for a controller to always seek convergence, as long as the system stays within a region from which it can converge within a finite number of steps. Capture regions are particularly useful in legged locomotion since they are typically easier to compute than viability kernels, and usually as large or nearly so [12], [27]. The core results of this thesis, namely [2], [3], can be reformulated to use capture regions instead of viable sets, without much loss of precision. I find, nonetheless, the more exact language of viability theory to be useful in formalizing the mathematical objects developed in this thesis.

2.2 Dynamic legged locomotion

While most results in my publications are valid for arbitrary dynamical systems, the motivation is grounded in legged locomotion, in particular running. This field offers abundant natural inspiration, including ourselves. Legged systems also encapsulate many interesting challenges in dynamics. The dynamics of legged systems are non-smooth, due to impacts at foot touchdown. They are hybrid, as the governing equations of motion switch abruptly every time a foot touches down or lifts off. They are underactuated, due to the floating base. While passively stable legged systems exist, for most systems of interest to us, they are passively unstable and require feedback control. And of course, they are typically highly nonlinear.

A core recurring concept in this field is the *natural dynamics* of the system: intuitively, how the system 'wants' to move. Controllers should be designed to exploit the natural dynamics, and robots should be designed to exhibit favorable natural dynamics. However, a rigorous definition of natural dynamics is elusive, especially for what constitutes more or less favorable natural dynamics. A core contribution of this thesis is to show that robustness to uncertainty is an important quality of 'favorable' natural dynamics, and provide a measure to quantify this.

This section introduces various ways in which the concept of natural dynamics comes into play, using compliance in running motion as the driving example.

2.2.1 Reduced order models of legged locomotion

Reduced order models allow researchers to cope with the complexity of legged locomotion dynamics. In this thesis, I will focus on the spring-loaded inverted pendulum (SLIP) model for running. This model originated in the biomechanics community to describe center-of-mass movement of running in humans [28]. Comprising of a point-mass to represent the body and a massless spring to represent the leg, the model has only a few parameters, making it relatively easy to fit to data. Several studies have then fit it to various animals of different sizes and with different numbers of legs [15], [29], [30]. A two-legged extension also accurately predicts ground-reaction forces of both running and walking in humans [31]. Daley and Biewener [15] used this model to explain the open-loop robustness to height perturbations observed in running birds. These studies of this parsimonious model convincingly show the presence of compliance in natural running motion.

It is no surprise that, in parallel to the development of this model in the biomechanics community, Raibert developed his famous hopping robots using very similar models and intuition [32, cf. Figure 2.5]. Since then, compliance has been frequently reproduced in bio-mimetic robots [33]–[38].

These simple models have also been used to study various control concepts [17], [39], [40]. In one of the most insightful examples, Wu and Geyer [41] show an open-loop trajectory for the swing-leg that achieves deadbeat control: any ground-height perturbation can be completely rejected in a single step.

However, since this model is energy-conservative and neglects most degrees of freedom, it is often insufficient to describe many observations of interest. For example, extensions that include actuation have been used to draw conclusions on control priorities [25], [42]. In another example, Maus, Revzen, *et al.* [24] use a data-driven approach to suggest various extended state and control laws for the SLIP, based on experimental observations of humans.

Reduced order models are also often used for mathematical analysis, as their low dimensional state space and parsimonious parameter space allow for thorough investigation of the dynamics and parameters. In my own master thesis work [43], I derive the explicit equations of motion (EoM) of a running model with a tail. This is made possible by approximating the tail as a flywheel, and allows insight on scaling effects of different parameters of the tail, which could then be tested directly in hardware.

The low dimensionality also allows numerical analysis by brute force. The basins of attraction for many such models have been studied numerically [17], [44]–[46], and extended in some cases to bifurcation analysis [16], [20], [47]. By making use of the Poincaré section, the stability of a limit-cycle can be numerically ascertained by Floquet analysis [48]. Reduced order models are also used by Byl and Tedrake [26] in their study of metastability, mentioned above. Since computing the mean first-passage time requires a very large number of simulations, it would be prohibitive for more sophisticated models.

2.2.2 Heuristic control for legged locomotion

Perhaps the most extreme examples of exploiting natural dynamics are the robots inspired by passive dynamic walkers [49]. These robots feature minimal sensing and actuation, and are typically limited to flat ground [50], [51]. They make up for their lack in maneuverability with extreme efficiency: the main purpose of the controller is not to stabilize the system, but to inject small amounts of energy to compensate for impact and friction losses. Tedrake, Zhang, *et al.* [6] saw in natural dynamics more than just the opportunity for efficient motion: they use a biped robot based on passive dynamic walkers, and show it can efficiently and reliably learn and adapt to changing ground surfaces. Formalizing this insight is one of the main motivations for this thesis.

Many of the earlier successful legged robots relied on exploiting natural dynamics by using simple controllers in the form of clocks and oscillators [18], [34], [52], [53]: motor positions are servoed along predetermined, periodic trajectories to the timing/phase of a clock/oscillator. All of these robots incorporate com-

pliance in the form of mechanical springs. Once tuned to the natural dynamics, the clock and oscillator controllers generate very stable and dynamic gaits. With some feedback, the oscillators also adapt to the natural dynamics, resulting in different gaits depending on the morphology [18] or driving frequency [18], [54]. However, from my experience and discussions with more seniors researchers in this field, these approaches rely on a lot of intuition and trial-and-error to design. They are particularly challenging to tune for unstable systems such as bipeds with point-feet.

Another approach is to use a hierarchical control scheme, in which the low-level controller causes the system to behave like a reduced order model, such as the SLIP model discussed above. If this low-level controller is accurate enough, the plethora of control laws studied for the SLIP model can be directly used as high-level controllers. To this end, there have been many efforts to design such a low-level controller [55]–[57]. Unlike for other models such as the linear inverted pendulum model, however, it is not trivial to map between the high-level representation of a SLIP model and low-level representation of an actual robot, and this approach has only seldomly been applied in practice [58]. Furthermore, these high-level controllers *exploit the natural dynamics of the reduced order model*, but not necessarily those of the system. It is entirely possible for the actual natural dynamics of the system to be negated by the low-level controllers. For these reasons, I believe reduced order models such as the SLIP are excellent *descriptive* models but poor *prescriptive* models: they allow a deep understanding, but make for poor control targets.

Virtual model control [59], [60] provides a slightly more relaxed approach: the low-level controller commands torques to mimic a spring-damper between two arbitrary points, usually between the hip and the foot. This approach doesn't force the system's dynamics onto the submanifold of the SLIP model but does retain the complaint behavior. It is also easy to implement, requiring only knowledge of the kinematics. However, this approach does not capture the impedance of the system, and it can be difficult to stabilize the robot trunk in more aggressive motion.

2.2.3 Optimal control for legged locomotion

Optimal control offers a more explicit approach to embed the natural dynamics in the controller. The setting is to find a policy $u_k = \pi(x_k)$ which minimizes the total cost-to-go function J given a dynamical system $x_{k+1} = f(x_k, u_k)$:

$$\begin{aligned} &\text{find } u_k = \pi(x_k) \\ &\text{such that } J^*(x_0) = \min_u \left[g_N(x_N) + \sum_{k=0}^{N-1} g(x_k, u_k) \right] \\ &\text{subject to } x_{k+1} = f(x_k, u_k) \\ &\qquad\qquad c(x,u) \leq 0 \end{aligned}$$

where J^* is the optimal cost-to-go accumulated between any state x_0 and a terminal state x_N, the final cost of being in x_N is g_N, and $g(x_k, u_k)$ is the cost incurred for applying the control input u_k from state x_k. This formulation explicitly takes the dynamics of the system into consideration, since the solution must be consistent with the dynamics $f(x_k, u_k)$. One of the core insights in optimal control is the *principle of optimality*: the fact that any subtrajectory of an optimal trajectory is itself optimal for its starting and ending states. Therefore, the optimal control problem can be split into several small steps, and each of these solved individually. This principle is leveraged in the Bellman equation (or its continuous-time equivalent, the Hamilton-Jacobi-Bellman equation):

$$J^*(x_k) = \min_u \left[g(x_k, u_k) + J^*(f(x_k, u_k)) \right]. \tag{2.1}$$

If the optimal cost-to-go function is known, finding an optimal controller is reduced to a one-step lookahead optimization. To find J^*, we can simply use the Bellman equation as an update rule iteratively, which leads to the *dynamic programming algorithm*. A typical iteration will start at the final state x_N, apply these updates by backing out from the final state. Once J^* has converged, a control policy that greedily follows eq. 2.1 will be optimal from any state. The main caveat with this approach is that it does not scale well with dimensionality.

Perhaps the most popular optimal control approach used in legged locomotion today is trajectory optimization: in this setting, we give up on global optimal-

ity and instead seek trajectories that are only locally optimal. For the interested reader, I recommend [61]. Without going into detail, there are two main families of trajectory optimization: shooting methods and direct transcription methods.

Shooting methods iterate making a forward pass, using an initial guess to simulate a trajectory, and then a backward pass to compute updates to the control inputs used. A popular and fast shooting method is differential dynamic programming [62], in which each backward pass updates the control trajectory with a Bellman update using a local, quadratic approximation of the cost-to-go function. This allows the updates to be backed out of the final state, in a conceptually similar way as the dynamic programming algorithm. Other shooting methods will typically pass the entire trajectory to a general nonlinear programming (NLP) solver, which optimizes over the entire trajectory at once [61]. This approach tends to struggle with problems that require constraint satisfaction and complex controls[1], such as legged locomotion. Nonetheless, it has been used successfully for offline design and analysis [48], [63].

Transcription methods break up the problem completely and allow an NLP solver to optimize not only over the control inputs at each time step, but also the states. Dynamic consistency is then enforced by transcribing the dynamics into constraints the solver must satisfy [61]. This formulation also allows constraints to be directly encoded in the optimization problem, which is not straight-forward for shooting methods.

A significant challenge for all these methods is the presence of contacts in legged locomotion. Making or breaking a contact induces a non-smooth jump in the cost, which solvers struggle with. While some formulations can directly reason about contacts [64]–[66], they tend to be slow. A common approach is to split the problem into parts that can be solved separately in a hierarchical control scheme. Typically, these will include generating a center of mass (and sometimes momentum) trajectory [65], [67], a sequence of footstep positions and tim-

[1] I will not attempt a rigorous definition of what more or less 'complex' controls are. Suffice it to say that legged locomotion typically requires controls which are more complex than orbital dynamics of satellites.

ings [67], [68], and a whole-body controller to execute the first two [67], [69]. Much of the current research aims at improving the robustness [70], [71], efficiency [72], [73], or flexibility [74] of these individual problems. While the viability-based tools presented in this thesis do not scale well, they can be used on these lower-dimensional sub-problems.

2.2.4 Hardware Design for legged locomotion

Just as we strive for controllers that can exploit a system's natural dynamics, we strive for hardware designs that feature beneficial natural dynamics. One such aspect, as discussed in Subsection 2.2.1, is compliance. We mentioned above several designs [18], [34], [52], [53] which use highly-geared motors and position control; the output of these 'stiff' actuators then drive mechanical springs, which provide the desired compliance. The springs are placed distally, and therefore minimize unsprung mass and protect the motors from harsh impacts. Additionally, simple controllers can achieve remarkable stability while running on inexpensive hardware with low update rates [75]. However, the complexity is traded off into the hardware design, and tuning hardware is usually much more time-consuming and expensive than tuning software. Furthermore, once built, it is usually difficult to change parameters without disassembling the robot. Since different behaviors often require very different natural dynamics, this approach can severely limit a robot's versatility.

A second approach is to incorporate the compliance directly in the actuators. Indeed, Raibert's first hoppers used pneumatic actuators, which act as air-springs in addition to injecting energy [32]. They require, however, a supply of pressurized air, which is cumbersome to carry on the robot. Series-elastic actuators (SEA) [76] are a popular alternative, and several of the most successful robots of today are built with this type of actuation, such as ANYMAL [36] or ATRIAS [35]. In these robots, the output of highly-geared motors is coupled in series with a relatively stiff spring. By measuring the position of both ends of the spring, the force output at the end-effector can be directly calculated. This allows force/torque control at the joints, and compliance of the desired form can be gen-

erated, within the limitations of the motor constraints and kinematics. The main drawbacks are added complexity for manufacturing and low-level control.

A third approach eschews any mechanical springs in favor of generating compliance purely through actuation and control. Low gear-ratios are required to minimize reflected inertia and achieve transparency while still maintaining high peak torque outputs [77]. Fortunately, the widespread success of quadcopters has dramatically reduced the cost of off-the-shelf outrunner motors that satisfy these requirements. Together with field-oriented control, sometimes called vector control, these motors provide an inexpensive solution at smaller scales. Many small and medium-sized robots capable of highly dynamic motion and direct torque-control are emerging, such as the MIT mini-cheetah [78], the Minitaur [37] and MPI-IS's own Solo [79]. Since mass scales roughly cubically[2], these smaller-scale robots are mechanically sturdy [80] and can operate at torques that are safe to handle. These robots are excellent experimental platforms, especially for learning control, since failures may be more common.

Regardless of the approach taken, design is still largely an art, and only a few specific aspects have been subject to rigorous study [81], [82]. One of the core contributions of this thesis is to formalize a quantification of robustness due to the system's natural dynamics [2], which will allow a designer to compare different designs objectively.

2.3 Learning control

The recent success in machine learning, in particular with reinforcement learning (RL), provides another alternative to model-based optimal control. My thesis work has been strongly influenced by ideas from reinforcement learning, especially the notion of state-action space. For readers less familiar with this field, I will introduce the essential topics needed to understand and appreciate this thesis.

[2]Mass will scale cubically to length if we assume isometric scaling and constant, uniform density. While robot designs break these assumptions, this serves as a good rule of thumb.

2.3.1 Reinforcement learning is model-free optimal control

At its core, reinforcement learning (RL) relies on the same key insight as dynamic programming and optimal control: the *principle of optimality*. For the interested reader, I recommend comparing two formulations of value iteration: the one introduced in chapter 4.4 of the classical RL textbook by Sutton and Barto [83], and the one introduced in chapter 5.3 of the classical optimal control textbook by Bertsekas [84]. Aside from notation and a minor detail[3], the equations are the same: they are updates based on the Bellman equation (eq. 2.1). The critical difference is the presence (or absence) of a model, which dictates how updates can be applied. In the model-based case, the transitions and their associated costs are known, and each iteration of the algorithm can apply an update at every state. In the model-free case, the cost of a specific transition can only be discovered through experience. A learning agent must therefore execute roll-outs, and can only update states visited during this roll-out.

Since the cost of a transition cannot be known by consulting a model, it is convenient to keep track of the value of state-action pairs, instead of just states. The value in state-action space is called the Q-function, and leads to one of the breakthroughs in RL: q-learning [83, cf. 6.5]. An important feature of this approach is that it leads to the optimal action without needing to specify the policy. From any given state, any optimal policy must output an element from the level set of optimal actions. I will use this insight in [2] to quantify the robustness of a system without specifying a policy.

2.3.2 The challenge of robot learning

Kober, Bagnell, *et al.* [85] have written a comprehensive overview of RL for robotics. Although slightly dated, the primary challenges ([85, cf. 8.1]) have not changed considerably. Many core challenges stem from the difficulty of obtaining samples from real-world experiments.

[3] In [83], there is a discount factor that is introduced a few pages after the initial formulation in [84].

Perhaps the most obvious difficulty is that, unlike in virtual systems, hardware experiments can only run at physical time, and are difficult to parallelize. Hardware experiments tend to require large amounts of space and funding; unlike for simulations, the cost per experiment does not decrease dramatically with scale. A more fundamental bottleneck is user time: a user typically needs to set up each run, reset it after failures, and maintain and repair the robots. For example, Levine, Pastor, *et al.* [86] used 14 robot arms, which executed roughly 1.7 million grasp attempts in the span of roughly six months. A policy in the form of a deep neural network trained on this data achieves a respectable failure rate of 10%. In comparison, Silver, Schrittwieser, *et al.* [87] trained a policy from scratch to play the board game go. During training, the learning agent generated 4.9 million game-plays, and reached a super-human level in roughly 40 hours. Both of these projects disposed of prodigious resources, far more than I believe should be required if learning control is to be useful. Yet the gap in achieved performance is staggering, and these two examples highlight the difficulty of gathering samples in robotics.

Models can help in two ways: by allowing learning in simulation, and by informing the designer of the appropriate structure to impose on the control policy.

Models allow simulations to be used to bootstrap the learning process. Collecting data in simulation is typically a lot cheaper than in hardware, and the policy can then be refined on the real robot. However, the transfer from simulation to the real-world is not always straight-forward. The models used in simulation are often insufficiently accurate, especially for legged locomotion [88], [89], and the learned policies often over-fit to these models. In other words, they find policies that may be optimal (or at least work reasonably well) but are not robust. This is commonly known as the *sim-to-real* problem.

A solution to this is dynamics randomization [89]–[91]: the parameters of the simulation are perturbed in each learning iteration, with the hope that the resulting control policy is not only robust to differences in the simulated model, but also in the real world. This approach shows promise. However, to the best of my knowledge, there is no principled approach to choose which parameters to perturb,

or by how much.

Another approach is to ensure your simulation is sufficiently accurate: Hwangbo, Lee, *et al.* [92] train a deep neural network to simulate the motor dynamics, which are typically difficult to model with first principles. A policy trained in simulation using this high-fidelity simulation, and transferred to the robot[4].

Model-based control can also be used directly for controller design, effectively warm-starting the learning process. In some of the most straightforward approaches, tools from conventional control theory and learning are combined but kept distinct. For example, Kumar, Ha, *et al.* [93] start by approximating a humanoid robot with the linear inverted pendulum model, and design a balance and stepping LQR controller based on this model. Then, a deep neural network is trained, which outputs offsets to the joint torques and target angles to enlarge the region of attraction. In another example, Yeganegi, Khadiv, *et al.* [71] start with a standard hierarchical control setup: a trajectory optimization problem is formulated using a linear inverted pendulum model to generate CoM trajectories, and a whole-body controller tracks these trajectories. However, how robustly these trajectories can be tracked depends largely on the coefficients of the cost-function used in the high-level controller. A Bayesian optimization scheme is then used to search for cost coefficients which result in trajectories that are both robust and fast.

Conventional control theory can also inform the design of proper policy parameterization. An appropriately chosen parameterization has a significant effect on the size of the search space as well as the resulting stability of the closed-loop system [94]. Indeed, directly learning a policy which outputs motor torques is often ineffective, especially for systems that are dynamically unstable [95]. In recent work, Viereck, Kozolinsky, *et al.* [96] train a policy in the form of a deep neural network which outputs trajectories of desired states and stabilizing feedback gains. These trajectories are more robust and reliable than directly learning a state-to-torque mapping [96]. In another example, Marco, Hennig, *et al.* [97]

[4]Dynamics randomization is also used in this study, but the core novelty is the use of supervised learning to obtain an accurate simulation.

incorporate the structure of a linear-quadratic regulator (LQR) into the kernel of a Gaussian process (GP). Extrapolation from previously observed samples is also more accurate.

A model-free approach to adding domain knowledge is to design an appropriately informative proxy reward function, which is used in lieu of the original reward function. This is particularly effective when the original reward function is *sparse*, meaning rewards are only experienced occasionally. In these cases, it is often difficult for the learning agent to properly attribute the correct value to the actions which were most meaningful. Though excessive massaging of a reward function can heavily bias the learning outcome, it also helps convergence tremendously by *shaping* the reward landscape [98], [99]. In effect, the purpose of this proxy reward function is to allow the learning agent to sample accurate gradient information more reliably.

Curriculum learning [1], [100], [101] is an effective compromise: the proxy reward function is only used temporarily. Once the learning agent has converged to a reasonable policy, it can continue learning without the assistance of the more informative reward function. While there is empirical evidence that this approach can be very effective, it also requires a lot of intuition and experience to design effective and general curricula.

There is great interest recently in automating the design of curricula, for example by modeling student progress as a separate optimization problem [102], or allowing the student direct control over the task difficulty [103]. The work of Kumar, Ha, *et al.* [93], mentioned earlier, also incorporates an adaptive curriculum. Perturbations are applied close to the edge of the currently known region of attraction of the humanoid robot. By collecting samples near the edge of stability, these samples are very likely to be both highly informative and remain safe. In another example, Ivanovic, Harrison, *et al.* [104] formally combine concepts from curriculum learning and backreachability to learn from a sparse reward signal effectively. In this case, the learning agent begins training near the target goal, such that even a random walk is likely to sample the reward. In subsequent iterations, the agent is initialized from a successively larger set of initial states. Instead of in-

creasing this set arbitrarily, it is increased by computing an approximate backward reachable set over a short time horizon. In this manner, the curriculum directly takes the dynamics of the system into consideration.

In all of these approaches, one of the most critical feedback loops is the designer, typically a graduate student. This outer feedback loop is described in substantial detail by Xie, Clary, *et al.* [105]: they describe many of the practical considerations required when learning on robots, and how these are accounted for in an iterative design of the learning problem. They successfully train a policy in simulation that they can deploy directly on the Cassie biped without any dynamics randomization.

Each of these approaches addresses the same fundamental issue: the need for robustness.

Chapter 3

Published Work

This cumulative thesis is based on the four first author, peer-reviewed publications I wrote during my studies. In this chapter, I provide some remarks on recommended reading orders for these papers, which are available in the appendix. For each paper, I then provide the abstract, commentary, some context of the publication venue[1] and individual contributions to each paper.

3.1 Overview

The recommended complete reading order is:

1. Shaping in Practice: Training Wheels to Learn Fast Hopping Directly in Hardware, *Steve Heim, Felix Ruppert, Alborz Sarvestani, and Alexander Spröwitz*, ICRA 2018

2. Beyond Basins of Attraction: Quantifying Robustness of Natural Dynamics *Steve Heim and Alexander Spröwitz*, T-RO 2019

3. A Learnable Safety Measure *Steve Heim, Alexander von Rohr, Sebastian Trimpe, and Alexander Spröwitz*, CoRL 2019

4. Learning from Outside the Viability Kernel: Why we should build Robots that can Fall with Grace *Steve Heim and Alexander Spröwitz*, SIMPAR 2019

[1] This includes my subjective opinion on the quality of publications published in this venue.

The first paper is an empirical study that captures much of the motivation. A 2-minute video synopsis of this paper available online at `https://youtu.be/6iH5E3LrYh8`, and can be viewed in lieu of reading the paper for an intuitive grasp of the results. The second paper formalizes viable sets in state-action space, which is the core theoretical contribution of the thesis. The third paper builds on the second to formalize model-free safe learning. For the most concise introduction to the core results, a reader may choose to start with this paper. The fourth paper is an offshoot of the second and points out a minor, counterintuitive result.

3.2 Papers and Contributions

For each paper, the contributions of co-authors are listed in table format, as well as a more detailed description of my contributions. Deciding on the contribution split is a subjective and imprecise exercise. I asked each student co-author to first consider independently how they thought contributions should be split. We then compared notes, discussed it together, and agreed to the description I have written here. This was presented to the principal investigator (PI) co-authors for their approval. To reflect the imprecise nature of these descriptions, I do not use a percentage split but rather a descriptive word: central, major, substantial, or minor. As a rule of thumb, **central** indicates a contribution that is central to the paper, **major** indicates a critical contribution, without which the paper would probably not have been submitted, **substantial** indicates an important contribution, without which the paper quality would be significantly diminished and possibly not accepted, and finally **minor** indicates contributions that improved the paper, but did not significantly change the form or content.

3.2.1 Shaping in practice

Abstract:
Learning instead of designing robot controllers can greatly reduce engineering effort required, while also emphasizing robustness. Despite considerable progress in simulation, applying learning directly in hardware is still challenging, in part

due to the necessity to explore potentially unstable parameters. We explore the concept of shaping the reward landscape with *training wheels*; temporary modifications of the physical hardware that facilitate learning. We demonstrate the concept with a robot leg mounted on a boom learning to hop fast. This proof of concept embodies typical challenges such as instability and contact, while being simple enough to empirically map out and visualize the reward landscape. Based on our results we propose three criteria for designing effective training wheels for learning in robotics.

Commentary:

One of the most important take-aways from this empirical study is that failures are common and typically provide very little useful information to learn from. When considering safety in robot learning, the usual motivations are avoiding damage (to the robot or the world) and time-consuming resets and repairs. Here we see, however, that it is also important in order to learn more reliably.

Venue:

The IEEE International Conference for Robotics and Automation (ICRA) is the largest and one of the most important conferences in robotics, alongside the International Conference on Intelligent Robots and Systems (IROS). Publications in these venues often have a high impact, and they are considered a top publication venue for the field of robotics. However, the variance on the quality of publications is also high, presumably due to the high variance in reviewing quality.

Individual Contributions:

I helped form the initial idea of demonstrating shaping with a mechanical change (training wheels), helped decide how to implement it, wrote the code for the locomotion controller, the learning algorithm, and code to process, analyze, and visualize the data. I conducted a large portion of experiments and wrote the paper, except for an excerpt describing the hardware. I took part in regular team meetings, and helped plan and adjust milestones throughout the project.

	Concept	**Implementation**	**Analysis**	**Submission**	**Management**
S.H.	Central	Central	Central	Central	Major
F.R.	Substantial	Central	Substantial	Major	Major
A.S.	Substantial	Central	Substantial	Substantial	Major
A.S. (PI)	Substantial	Minor	Minor	Substantial	Major

3.2.2 Beyond basins of attraction

Abstract:
Properly designing a system to exhibit favorable natural dynamics can greatly simplify designing or learning the control policy. However, it is still unclear what constitutes favorable natural dynamics and how to quantify its effect. Most studies of simple walking and running models have focused on the basins of attraction of passive limit-cycles and the notion of self-stability. We instead emphasize the importance of stepping beyond basins of attraction. We show an approach based on viability theory to quantify robust sets in state-action space. These sets are valid for the family of all robust control policies, which allows us to quantify the robustness inherent to the natural dynamics before designing the control policy or specifying a control objective. We illustrate our formulation using spring-mass models, simple low dimensional models of running systems. We then show an example application by optimizing robustness of a simulated planar monoped, using a gradient-free optimization scheme. Both case studies result in a nonlinear effective stiffness providing more robustness.

Commentary:
To the best of my knowledge, this is the first rigorous formalization of how morphology or mechanical design influences robustness. The formalization of viable sets in state-action space allows quantified comparisons of different designs in terms of robustness, without making assumptions on the control policy design. Furthermore, focusing on robustness in terms of avoiding failure instead of convergence is a subtle yet important change in how we reason about locomotion. Although the fundamental mathematical objects are defined in this paper, there is

a lot of room for future work in applying these objects in different contexts and for specific systems. Another open topic is computational tools to apply this in practice.

Venue:

Transactions on Robotics is the flagship journal of the IEEE Robotics and Automation Society. It is widely considered one of the top journals in robotics, either on par or a close second to the International Journal on Robotics Research (IJRR).

Individual Contributions:

I formed the concepts, developed them via numerical experimentation, and formalized these concepts mathematically. I refined the direction based on feedback to preliminary numerical results at conferences such as 'Dynamic Walking' and 'Adaptive Motion in Animals and Machines'. I devised the algorithms, wrote all code for simulation, numerical computation, and visualization. I wrote the paper, and responses to reviewers, and managed my time and goals on this project, including splitting off the side result to a separate publication (publication 4).

	Concept	**Implementation**	**Analysis**	**Submission**	**Management**
S.H.	Central	Central	Central	Central	Central
A.S. (PI)	Minor	Minor	Minor	Substantial	Substantial

3.2.3 A learnable safety measure

Abstract:

Failures are challenging for learning to control physical systems since they risk damage, time-consuming resets, and often provide little gradient information. Adding safety constraints to exploration typically requires a lot of prior knowledge and domain expertise. We present a safety measure which implicitly captures how the system dynamics relate to a set of failure states. Not only can this measure be used as a safety function, but also to directly compute the set of safe state-action pairs. Further, we show a model-free approach to learn this measure by active sampling using Gaussian processes. While safety can only be guaranteed after learning the safety measure, we show that failures can already be greatly

reduced by using the estimated measure during learning.

Commentary:

This paper builds on the mathematical objects introduced in *Beyond Basins of Attraction* [2] by placing a measure over the viable sets in state-action space. The primary motivation of the paper is to allow learning directly on the real system, by restricting exploration to probabilistically safe regions. A side effect is that samples are restricted to 'interesting' regions: as pointed out in several papers in this thesis, failures are not only costly due to potential damage, but they are also typically uninformative for the learning process. By concentrating samples in the informative, non-failing set, the learning agent wastes fewer samples. Beyond the robot learning setting, this approach can be used in other contexts where large portions of the search space are uninformative by simply replacing the notion of 'failure set' with an appropriate state-constraint.

Venue:

The Conference on Robot Learning (CoRL) is a young conference: this publication appears in its third edition. It is loosely modeled after the Robotics Science and Systems (RSS) conference, another small, single-track conference known for its high-quality and low acceptance rate; in the third edition of CoRL, 112 papers were selected out of 398 submissions. Papers from previous editions quite consistently have high citation counts after only 1-2 years. This high citation count can be partly attributed to the high standards of the conference, and partly to the current popularity (and therefore high publication rate) of machine learning.

Individual Contributions:

I formed the concept and developed the theoretical groundwork in a deterministic setting, including the convergence proof. I wrote the code framework for computing viable sets, as well as all simulations. I recognized the possibility of extending this to a probabilistic setting, and reached out to Alexander von Rohr to help on this aspect. I helped work out the details of modeling the measure as a Gaussian process, and how to use this probabilistic model to gather samples. I helped debug and test the learning algorithm and tune hyperparameters. I wrote the majority of the paper, proof-read the paper, and helped make visualizations. I managed the

milestones and overall timeline for the project and helped plan milestones for the project.

	Concept	**Implementation**	**Analysis**	**Submission**	**Management**
S.H.	Central	Central	Central	Central	Central
A.v.R.	Substantial	Central	Central	Major	Major
S.T. (PI)	Minor	Minor	Minor	Substantial	Substantial
A.B.S. (PI)	Minor	Minor	Minor	Minor	Substantial

3.2.4 Learning from outside the viability kernel

Abstract:

Despite impressive results using reinforcement learning to solve complex problems from scratch, in robotics this has still been largely limited to model-based learning with very informative reward functions. One of the major challenges is that the reward landscape often has large patches with no gradient, making it difficult to sample gradients effectively. We show here that the robot state-initialization can have a more important effect on the reward landscape than is generally expected. In particular, we show the counterintuitive benefit of including initializations that are *unviable*, in other words initializing in states that are doomed to fail.

Commentary:

I find the core result of this work interesting since it is particularly counterintuitive, and it gives a simple, concrete example that highlights why it is a good practice to place a lot more variation in state initializations. This aspect is very often completely ignored when designing experimental set-ups.

Venue:

The IEEE international conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR) is a small (27 accepted papers that year), bi-annual conference typically organized in proximity of a large robotics conference: ICRA for this year. The theme for this edition was 'Leveraging Simulation in the Hardware and Software Design of Autonomous Robots'. The overall quality of

publications appeared to me to be decent, neither particularly high nor low.

Individual Contributions:

I formed and developed the concept, wrote the code and carried out experiments and analysis. I chose the venue of publication, wrote the paper, and managed work packages and milestones for the project.

	Concept	**Implementation**	**Analysis**	**Submission**	**Management**
S.H.	Central	Central	Central	Central	Central
A.S. (PI)	Minor	Minor	Minor	Substantial	Substantial

Chapter 4

Discussion

For a detailed discussion specific to the results of each paper, we refer to the papers themselves. Here we will discuss the common themes, the connections between them, and what the results of this thesis mean for future work.

4.1 Designing robots that can avoid failure

As demonstrated in [1], robustness to failures is critical to learning control, and appropriate system design can greatly influence the inherent system robustness. Our work in [2] formalizes what a design should strive for: robust natural dynamics.

I show an example of computational design in [2]. However, it is limited to optimizing the parameters of a low-level control for a relatively simple legged system. To truly make use of these concepts for computational design (both of robot hardware and low-level controls) will require further progress in scaling the tools to compute viable sets in state-action space.

The work in [3] begins to address this: although the main contribution of the paper is to enable safe learning directly on a robot, it can also be used in simulation to learn approximations of viable sets, and scales better than the brute-force computation used in [2]. Nonetheless, further tools to allow computational tractability are needed to make these concepts useful for higher-dimensional systems. Ex-

ploiting structure in well-understood dynamical systems, such as the rigid-body dynamics commonly seen in robots, is a promising path. How this structure relates to the set-valued dynamics view of viability is an open question.

In parallel to developing computational tools, I also see value in the intuition gleaned from reduced order models. Indeed, armed with the knowledge that robustness to noisy action spaces will make controller design easier, a robot designer can already make more deliberate design decisions, guided by this intuition.

4.2 Designing robots that can fail

The work of this thesis also emphasizes the need for robots to be designed such that occasional failures are not critical. The most important reason for this is that guaranteeing safety requires perfect models, which, of course, do not exist [5]. The computational tractability of sufficiently accurate models is, as discussed above, still a major challenge. It is then only reasonable to begin with simpler, less accurate but more tractable models, before transferring directly to the real system. To use these less accurate results in practice, we can take two approaches.

One approach is to compute under-approximated viable sets, which will retain guarantees of safe operation. However, these tools also difficult to scale to higher dimensions, and often result in over-conservative approximations [106].

A different approach, which is mentioned in [3], is to use data from actual operation of the robot to refine an initial simulation-driven approximation of the viable set. This approach is contingent on robots being allowed to fail occasionally. In practice, this means a robot should be sturdy enough to survive a few failures, but also that a failure does not cause unacceptable damage to the user or environment. It is also useful if it is possible to reset the robot quickly and easily. Just as in curriculum learning, it may be possible to gather at least a portion of these data samples in a controlled environment: a sandbox, so to speak, where failures are tolerable.

A second reason why robots should be designed to allow failures is illustrated in [4]. The core concept is that systems often have large sets of unviable states:

states which have not failed yet, but are doomed to fail within finite time. In this time, the learning agent can still explore and collect informative samples that are useful for the learning process — if it can survive the failure. For this to be useful, two requirements should be fulfilled: first, information relevant to the learning process should be available in unviable states. This will depend largely on the reward function. Second, a well-designed system will not fail abruptly, but rather be able to continue exploration for as long as possible, delaying the inevitable failure. Fortunately, this requirement often goes hand in hand with improving robustness, as discussed above. For example, starting with soft gains on an impedance controller will allow a robot to stumble and fall more slowly than a controller that is aggressively tuned for performance.

4.3 Leveraging dynamics models

I have focused on model-free methods, for a very deliberate reason: I believe that model-based optimal control is a fantastically powerful tool, and it is too tempting to try to reformulate any problem as an optimization problem. Many of the results on robustness developed in this thesis, although not mathematically complicated, are not evident through the lens of optimality. Taking a completely (or nearly) model-free approach forces us to deal directly with robustness, as we have no assumed structure to exploit. It has strongly motivated the development of the new, mathematically simple objects presented in [2], and the clean simplicity for safe learning in [3]. Now that the fundamental mathematical objects have been established, the time is ripe for turning to model-based tools. Leveraging this structure will be an essential key to scaling up these concepts to use models in higher dimensions.

Several computational tools are being developed to compute invariant sets in state-space. For example, sums-of-squares polynomials can be cast as semi-definite programs (SOS programming), and have been used to compute robust funnels [107] and robust regions of attraction [108]. Although these tools are typically used for analyzing convergence in the Lyapunov sense, it is reasonable to

expect the same tools to work well for viable sets, which are also invariant sets.

Because these tools are computationally rather demanding, they have mostly been used for analysis. However, recent developments are pushing their utility for control, for example by combining SOS programming with reachability analysis in a hierarchical framework [109], or combining it with trajectory optimization [70]. Similar combinations with a view of viability in state-action space have, in my opinion, a lot of potential and offer exciting opportunities.

Directly learning a model of the dynamics while also learning a model of the safety measure is, in my opinion, one of the most promising and straightforward next steps. In particular, each of these learning processes can benefit from the other in terms of sample efficiency: on the one hand, it is desirable to focus sampling in viable regions when gathering samples for learning dynamics. On the other hand, a model of the dynamics can be used to bootstrap learning the safety measure. Furthermore, predictions can help active sampling by 'checking ahead' to compare the safety of a state-action pair with the safety of the predicted next state. How to trade-off between uncertainty of the prediction compared to the model of the safety measure is something I hope to look into soon.

Bibliography

[1] S. Heim, F. Ruppert, A. A. Sarvestani, and A. Spröwitz, "Shaping in practice: Training wheels to learn fast hopping directly in hardware," in © *2018 IEEE Reprinted, with permission, from International Conference on Robotics and Automation (ICRA)*, pp. 5076–5081. DOI: 10.1109/ICRA.2018.8460984.

[2] S. Heim and A. Spröwitz, "Beyond basins of attraction: Quantifying robustness of natural dynamics," © *2019 IEEE Reprinted, with permission, from Transactions on Robotics*, vol. 35, no. 4, pp. 939–952, ISSN: 1941-0468. DOI: 10.1109/TRO.2019.2910739.

[3] S. Heim, A. von Rohr, S. Trimpe, and A. Badri-Spröwitz, "A learnable safety measure," in *Proceedings of The 3rd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, PMLR, 2019.

[4] S. Heim and A. Spröwitz, "Learning from outside the viability kernel: Why we should build robots that can fall with grace," in © *2018 IEEE Reprinted, with permission, from International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pp. 55–61. DOI: 10.1109/SIMPAR.2018.8376271.

[5] G. E. Box, "Science and statistics," *Journal of the American Statistical Association*, vol. 71, no. 356, pp. 791–799, 1976.

[6] R. Tedrake, T. W. Zhang, and H. S. Seung, "Learning to walk in 20 minutes," in *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University New Haven, vol. 95585, 2005, pp. 1939–1412.

[7] J. Randløv, "Shaping in reinforcement learning by changing the physics of the problem.," in *ICML*, 2000.

[8] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre, *Viability theory: new directions*. Springer Science & Business Media, 2011.

[9] S. H. Strogatz, *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC Press, 2018.

[10] S. Bazzi, J. Ebert, N. Hogan, and D. Sternad, "Stability and predictability in human control of complex objects," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, no. 10, p. 103 103, 2018.

[11] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec. 2017, pp. 2242–2253. DOI: 10.1109/CDC.2017.8263977.

[12] P. Zaytsev, W. Wolfslag, and A. Ruina, "The boundaries of walking stability: Viability and controllability of simple models," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 336–352, Apr. 2018. DOI: 10.1109/TRO.2017.2782818.

[13] A. Liniger and J. Lygeros, "Real-time control for autonomous racing based on viability theory," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 2, pp. 464–478, Mar. 2019. DOI: 10.1109/TCST.2017.2772903.

[14] G. Deffuant, L. Chapel, and S. Martin, "Approximating viability kernels with support vector machines," *IEEE transactions on automatic control*, vol. 52, no. 5, 2007.

[15] M. A. Daley and A. A. Biewener, "Running over rough terrain reveals limb control for intrinsic stability," *Proceedings of the National Academy of Sciences*, vol. 103, no. 42, 2006.

[16] A. Merker, D. Kaiser, A. Seyfarth, and M. Hermann, "Stable running with asymmetric legs: A bifurcation approach," *International Journal of Bifurcation and Chaos*, vol. 25, no. 11, p. 1 550 152, 2015.

[17] T. Cnops, Z. Gan, and C. D. Remy, "The basin of attraction for running robots: Fractals, multistep trajectories, and the choice of control," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1586–1591.

[18] D. Owaki, T. Kano, K. Nagasawa, A. Tero, and A. Ishiguro, "Simple robot suggests physical interlimb communication is essential for quadruped walking," *Journal of The Royal Society Interface*, vol. 10, no. 78, 2013.

[19] S. Aoi, D. Katayama, S. Fujiki, N. Tomita, T. Funato, T. Yamashita, K. Senda, and K. Tsuchiya, "A stability-based mechanism for hysteresis in the walk–trot transition in quadruped locomotion," *Journal of The Royal Society Interface*, vol. 10, no. 81, p. 20 120 908, 2013.

[20] Z. Gan, Y. Yesilevskiy, P. Zaytsev, and C. D. Remy, "All common bipedal gaits emerge from a single passive model," *Journal of The Royal Society Interface*, vol. 15, no. 146, p. 20 180 455, 2018.

[21] R. Wheatley, J. Angilletta Michael J., A. C. Niehaus, and R. S. Wilson, "How fast should an animal run when escaping? an optimality model based on the trade-off between speed and accuracy," *Integrative and Comparative Biology*, vol. 55, no. 6, pp. 1166–1175, Aug. 2015, ISSN: 1540-7063. DOI: 10.1093/icb/icv091.

[22] T. Y. Moore, K. L. Cooper, A. A. Biewener, and R. Vasudevan, "Unpredictability of escape trajectory explains predator evasion ability and microhabitat preference of desert rodents," *Nature communications*, vol. 8, no. 1, p. 440, 2017.

[23] S. Bruijn, O. Meijer, P. Beek, and J. Van Dieën, "Assessing the stability of human locomotion: A review of current measures," *Journal of the Royal Society Interface*, vol. 10, no. 83, p. 20 120 999, 2013.

[24] H.-M. Maus, S. Revzen, J. Guckenheimer, C. Ludwig, J. Reger, and A. Seyfarth, "Constructing predictive models of human running," *Journal of The Royal Society Interface*, vol. 12, p. 20 140 899, 2015.

[25] A. V. Birn-Jeffery, C. M. Hubicki, Y. Blum, D. Renjewski, J. W. Hurst, and M. A. Daley, "Dont break a leg: Running birds from quail to ostrich prioritise leg safety and economy on uneven terrain," *Journal of Experimental Biology*, vol. 217, no. 21, pp. 3786–3796, 2014, ISSN: 0022-0949. DOI: 10.1242/jeb.102640.

[26] K. Byl and R. Tedrake, "Metastable walking machines," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 1040–1064, 2009.

[27] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[28] R. Blickhan, "The spring-mass model for running and hopping," *Journal of biomechanics*, vol. 22, pp. 1217–1227, 1989.

[29] R. Blickhan and R. Full, "Similarity in multilegged locomotion: Bouncing like a monopode," *Journal of Comparative Physiology A*, vol. 173, no. 5, pp. 509–517, 1993.

[30] D. L. Jindrich and R. J. Full, "Dynamic stabilization of rapid hexapedal locomotion," *Journal of Experimental Biology*, vol. 205, pp. 2803–2823, 2002.

[31] 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*, vol. 273, no. 1603, pp. 2861–2867, 2006.

[32] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[33] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, 2013. DOI: 10.1177/0278364913489205.

[34] J. Buchli, F. Iida, and A. J. Ijspeert, "Finding resonance: Adaptive frequency oscillators for dynamic legged locomotion," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 3903–3909. DOI: 10.1109/IROS.2006.281802.

[35] C. Hubicki, J. Grimes, M. Jones, D. Renjewski, A. Spröwitz, A. Abate, and J. Hurst, "Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot," *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 12, pp. 1497–1521, 2016. DOI: 10.1177/0278364916648388.

[36] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[37] G. Kenneally, A. De, and D. E. Koditschek, "Design principles for a family of direct-drive legged robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, 2016.

[38] J. Ramos, B. Katz, M. Y. M. Chuah, and S. Kim, "Facilitating model-based control through software-hardware co-design," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 566–572.

[39] G. Piovan and K. Byl, "Two-element control for the active slip model," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 5656–5662.

[40] ——, "Reachability-based control for the active slip model," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 270–287, 2015. DOI: 10.1177/0278364914552112.

[41] A. Wu and H. Geyer, "The 3-d spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1114–1124, 2013.

[42] Y. Blum, H. R. Vejdani, A. V. Birn-Jeffery, C. M. Hubicki, J. W. Hurst, and M. A. Daley, "Swing-leg trajectory of running guinea fowl suggests task-level priority of force regulation rather than disturbance rejection," *PloS one*, vol. 9, no. 6, e100399, 2014.

[43] S. W. Heim, M. Ajallooeian, P. Eckert, M. Vespignani, and A. J. Ijspeert, "On designing an active tail for legged robots: Simplifying control via decoupling of control objectives," *Industrial Robot: An International Journal*, vol. 43, no. 3, 2016.

[44] A. Schwab and M. Wisse, "Basin of attraction of the simplest walking model," in *Proceedings of the ASME design engineering technical conference*, vol. 6, 2001.

[45] I. Obayashi, S. Aoi, K. Tsuchiya, and H. Kokubu, "Formation mechanism of a basin of attraction for passive dynamic walking induced by intrinsic hyperbolicity," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 472, no. 2190, p. 20 160 028, 2016.

[46] J. Rummel and A. Seyfarth, "Stable running with segmented legs," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 919–934, 2008.

[47] S. Aoi and K. Tsuchiya, "Bifurcation and chaos of a simple walking model driven by a rhythmic signal," *International Journal of Non-Linear Mechanics*, vol. 41, no. 3, pp. 438–446, 2006.

[48] C. D. Remy, K. Buffinton, and R. Siegwart, "A matlab framework for efficient gait creation," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2011, pp. 190–196.

[49] T. McGeer, "Passive dynamic walking," *The International Journal Robotic Research*, vol. 9, pp. 62–82, 1990.

[50] P. A. Bhounsule, J. Cortell, and A. Ruina, "Design and control of ranger: An energy-efficient, dynamic walking robot," in *Adaptive Mobile Robotics*, World Scientific, 2012, pp. 441–448.

[51] M. Wisse and J. Van Frankenhuyzen, "Design and construction of mike; a 2-d autonomous biped based on passive dynamic walking," in *Adaptive motion of animals and machines*, Springer, 2006, pp. 143–154.

[52] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, 2013.

[53] R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, H. Brown, D. McMordie, U. Saranli, R. Full, and D. E. Koditschek, "Rhex: A biologically inspired hexapod runner," *Autonomous Robots*, vol. 11, no. 3, 2001.

[54] D. Owaki and A. Ishiguro, "A quadruped robot exhibiting spontaneous gait transitions from walking to trotting to galloping," *Scientific reports*, vol. 7, no. 1, p. 277, 2017.

[55] M. Hutter, C. D. Remy, M. A. Höpflinger, and R. Siegwart, "Slip running with an articulated robotic leg," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2010, pp. 4934–4939.

[56] 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*, vol. 54, no. 8, pp. 1779–1793, 2009.

[57] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 5134–5140. DOI: 10.1109/IROS.2013.6697099.

[58] W. C. Martin, A. Wu, and H. Geyer, "Experimental evaluation of deadbeat running on the atrias biped," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1085–1092, 2017.

[59] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001.

[60] D. Renjewski, A. Spröwitz, A. Peekema, M. Jones, and J. Hurst, "Exciting engineered passive dynamics in a bipedal robot," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1244–1251, 2015.

[61] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[62] Y. Tassa, *Theory and Implementation of Biomimetic Motor Controllers*. Hebrew University of Jerusalem, 2011.

[63] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 3, pp. 321–330, 2009. DOI: 10.1017/S0263574708004724.

[64] I. Mordatch, E. Todorov, and Z. Popovi, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[65] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE International Conference on Humanoid Robots*, 2014.

[66] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, 2018.

[67] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *2009 IEEE/RSJ international conference on Intelligent robots and systems*, 2009, pp. 814–820.

[68] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE-RAS International Conference on Humanoid Robots*, Nov. 2014, pp. 279–286. DOI: 10.1109/HUMANOIDS.2014.7041373.

[69] L. Sentis, "Synthesis and control of whole-body behaviors in humanoid systems," Ph.D. dissertation, 2007.

[70] Z. Manchester and S. Kuindersma, "Robust direct trajectory optimization using approximate invariant funnels," *Autonomous Robots*, vol. 43, no. 2, pp. 375–387, 2019.

[71] M. H. Yeganegi, M. Khadiv, S. A. A. Moosavian, J.-J. Zhu, A. Del Prete, and L. Righetti, "Robust humanoid locomotion using trajectory optimization and sample-efficient learning," *arXiv preprint arXiv:1907.04616*, 2019.

[72] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5776–5782. DOI: 10.1109/ICRA.2018.8460537.

[73] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 5280–5286.

[74] C. Boussema, M. J. Powell, G. Bledt, A. J. Ijspeert, P. M. Wensing, and S. Kim, "Online gait transitions and disturbance recovery for legged robots via the feasible impulse set," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1611–1618, Apr. 2019, ISSN: 2377-3774. DOI: 10.1109/LRA.2019.2896723.

[75] A. T. Spröwitz, A. Tuleu, M. Ajallooeian, M. Vespignani, R. Möckel, P. Eckert, M. D'Haene, J. Degrave, A. Nordmann, B. Schrauwen, J. Steil, and A. J. Ijspeert, "Oncilla robot: A versatile open-source quadruped research robot with compliant pantograph legs," *Frontiers in Robotics and AI*, vol. 5, p. 67, 2018. DOI: 10.3389/frobt.2018.00067.

[76] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, IEEE, vol. 1, 1995, pp. 399–406.

[77] S. Seok, A. Wang, D. Otten, and S. Kim, "Actuator design for high force proprioceptive control in fast legged locomotion," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 1970–1975. DOI: 10.1109/IROS.2012.6386252.

[78] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 6295–6301.

[79] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, J. Fiene, *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *arXiv preprint arXiv:1910.00093*, 2019.

[80] A. A. Biewener, "Biomechanical consequences of scaling," *Journal of Experimental Biology*, vol. 208, no. 9, pp. 1665–1676, 2005.

[81] A. Abate, R. L. Hatton, and J. Hurst, "Passive-dynamic leg design for agile robots," in *2015 IEEE international conference on robotics and automation (ICRA)*, 2015, pp. 4519–4524.

[82] A. Abate, J. Hurst, and R. Hatton, "Mechanical antagonism in legged robots," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, Jun. 2016. DOI: 10.15607/RSS.2016.XII.008.

[83] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction, 2nd Edition*. MIT press Cambridge, 2019, vol. 1. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html.

[84] D. Bertsekas, *Dynamic Programming and Optimal Control, 4th Edition*. Athena Scientific, 2017, vol. 1. [Online]. Available: http://www.athenasc.com/dpbook.html.

[85] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[86] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[87] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, 2017.

[88] M. Neunert, T. Boaventura, and J. Buchli, "Why off-the-shelf physics simulators fail in evaluating feedback controller performance-a case study for quadrupedal robots," in *Advances in Cooperative Robotics*, World Scientific, 2017, pp. 464–472.

[89] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," 2018. DOI: 10.15607/RSS.2018.XIV.010.

[90] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *arXiv preprint arXiv:1710.06537*, 2017, Accepted to Robotics and Automation (ICRA), IEEE International Conference on.

[91] K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov, "Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system," in *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, IEEE, 2018, pp. 35–42.

[92] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019. DOI: 10.1126/scirobotics.aau5872.

[93] V. C. Kumar, S. Ha, and K. Yamane, "Improving model-based balance controllers using reinforcement learning and adaptive sampling," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7541–7547.

[94] J. W. Roberts, I. R. Manchester, and R. Tedrake, "Feedback controller parameterizations for reinforcement learning," in *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, IEEE, 2011, pp. 310–317.

[95] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 2017, p. 12.

[96] J. Viereck, J. Kozolinsky, A. Herzog, and L. Righetti, "Learning a structured neural network policy for a hopping task," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4092–4099, 2018.

[97] A. Marco, P. Hennig, S. Schaal, and S. Trimpe, "On the design of lqr kernels for efficient controller learning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 5193–5200.

[98] V. Gullapalli and A. G. Barto, "Shaping as a method for accelerating reinforcement learning," in *Intelligent Control, 1992., Proceedings of the IEEE International Symposium on*, 1992.

[99] J. Randlov and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[100] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009.

[101] A. Karpathy and M. Van De Panne, "Curriculum learning for motor skills," *Advances in Artificial Intelligence*, 2012.

[102] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer, "Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments," Proceedings of Machine Learning Research, 2019.

[103] P. Klink, H. Abdulsamad, B. Belousov, and J. Peters, "Self-paced contextual reinforcement learning," Proceedings of Machine Learning Research, 2019.

[104] B. Ivanovic, J. Harrison, A. Sharma, M. Chen, and M. Pavone, "Barc: Backward reachability curriculum for robotic reinforcement learning," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 15–21.

[105] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," Proceedings of Machine Learning Research, 2019.

[106] I. R. Manchester, M. M. Tobenkin, M. Levashov, and R. Tedrake, "Regions of attraction for hybrid limit cycles of walking robots," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 5801–5806, 2011.

[107] A. Majumdar and R. Tedrake, "Robust online motion planning with regions of finite time invariance," in *Algorithmic Foundations of Robotics X*, Springer, 2013, pp. 543–558.

[108] G. Valmorbida and J. Anderson, "Region of attraction analysis via invariant sets," in *2014 American Control Conference*, Jun. 2014, pp. 3591–3596. DOI: 10.1109/ACC.2014.6859263.

[109] S. Singh, M. Chen, S. Herbert, C. Tomlin, and M. Pavone, "Robust tracking with model mismatch for fast and safe planning: An sos optimization approach," 2018.

Appendix A

Published Papers

Shaping in Practice: Training Wheels to Learn Fast Hopping Directly in Hardware

Steve Heim, Felix Ruppert, Alborz A. Sarvestani, Alexander Spröwitz
Dynamic Locomotion Group
Max Planck Institute for Intelligent Systems, Germany

***Abstract*— Learning instead of designing robot controllers can greatly reduce engineering effort required, while also emphasizing robustness. Despite considerable progress in simulation, applying learning directly in hardware is still challenging, in part due to the necessity to explore potentially unstable parameters. We explore the concept of shaping the reward landscape with *training wheels*; temporary modifications of the physical hardware that facilitate learning. We demonstrate the concept with a robot leg mounted on a boom learning to hop fast. This proof of concept embodies typical challenges such as instability and contact, while being simple enough to empirically map out and visualize the reward landscape. Based on our results we propose three criteria for designing effective training wheels for learning in robotics. A video synopsis can be found at `https://youtu.be/6iH5E3LrYh8`.**

I. INTRODUCTION

In nature, animals learn to move with a grace and agility that is the envy of robotics engineers. One major challenge is that most algorithms rely on accurate models, which in turn also take a lot of engineering effort. Alternatively, reinforcement learning (RL) is a powerful paradigm that can work both model-based or *model-free*. In addition, reinforcement learning is often able to learn from generic and even highly delayed reward signals: for example a legged robot might receive a reward for reaching a specific target location within a set time limit, and no reward for getting progressively closer. This allows for easy and intuitive assignment of rewards without constraining the behavior for achieving the goal. Despite these attractive features and promising achievements in simulation [1][2], applying RL directly in hardware has proven challenging [3][4][5] with only a handful of successes that actually run model-free [6][7].

One major challenge in hardware comes from the necessity to *explore* the reward landscape. The landscape is usually non-convex, and often only subsets represent behaviors that actually accumulate reward: the rest of the landscape often looks flat, representing different behaviors that all receive the same or even no reward. Sampling from these regions provides no gradient information for the robot to learn from. This is particularly true when the reward is generic and delayed such as in the previous example: a policy that causes the robot to fall over immediately would get the same reward of zero as a policy that hops in place, even though the second policy arguably solves part of the locomotion problem [8]. This problem is even more accentuated in robots that are unstable, since instabilities often quickly lead to direct failure states. These failures generally lead to no reward, and can also damage the hardware. In practice, exploration is executed cautiously, usually locally. This combination means that large parts of the reward landscape are flat, and there is no salient gradient to lead the learning agent in the correct direction.

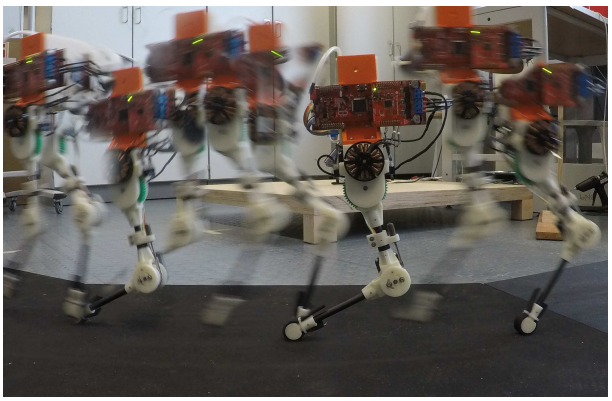


Fig. 1: A multiple-exposure image of our robotic leg mounted on a boom hopping, showing the distinct stance and flight phases.

The exploration challenge can be solved by choosing a more appropriate policy parameterization, or with different exploration strategies such as intrinsic motivation [9]. This can however be difficult to find and does not eliminate the flat regions, or the potentially damaging failures.

Another approach is to shape the reward landscape. A common method of shaping is to encode more information of the task in the reward [10]. The drawback is that it requires more prior knowledge of the task, and goes against the attractiveness of being able to choose rewards based on achieving a task rather than specifying a behavior[1]. It is also possible to shape the landscape by proper mechanical design. For example, walking robots designed after passive-dynamic walkers [11] have good stability properties for a wide range of policy parameters, allowing quick and reliable learning from even poor initializations [6]. The drawback is that designing the system around one specific behavior can

contact: lastname@is.mpg.de
Dynamic Locomotion Group, Max Planck Institute for Intelligent Systems, Heisenbergstr. 3, 70569 Stuttgart, Germany

[1]Whether the robot crawls, walks or runs should depend on the context of the situation and not on the goal.

be limiting in terms of versatility and design options.
We build on these ideas with the concept of *training wheels*: shaping the landscape with *temporary mechanical modifications of the robot that allow for easier learning*. To the best of our knowledge, this concept has only been briefly explored in simulation, even though initial results showed promise [12]. We present a proof of concept directly in hardware, applied to learning fast hopping of a monoped robot with a rolling foot: an underactuated, unstable system featuring hybrid dynamics.
We would like to note that we largely use the terminology of the RL community. In particular, the term *environment* signifies everything that is beyond direct control of the learning agent. Take for example an agent whose policy outputs a desired joint position; then the environment includes not only the physical world the robot moves in, but also the robot itself and the PD motor controller used to track the desired joint position. For a more thorough treatment of RL see [5][13].

II. SETUP: MECHANICS, POLICY AND LEARNING SCHEME

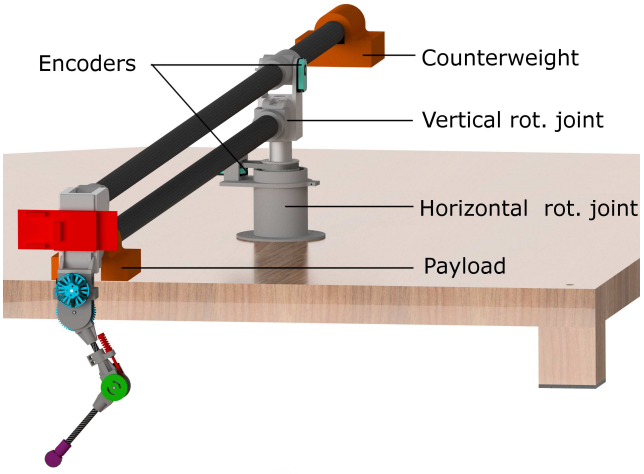


Fig. 2: The entire robot consists of a leg mounted on a boom, with a total of four degrees of freedom. The counterweight balances out the mass of the boom without the leg or payload. The payload represents the mass of the batteries and additional electronics, which are offloaded via a tether.

Our robot platform (Fig. 2) consists of a two-segment leg with a passive compliant ankle joint and an actuated hip joint, mounted on a boom which constrains the body to motion on a 2D surface. The robot thus has four degrees of freedom (DoFs) and a single actuator. The passive compliance at the ankle joint (Fig. 3) results in favorable natural dynamics [14], though the system is still passively unstable.
The learning task is to achieve fast hopping, and the reward for each rollout is the average speed with one additional condition: potentially damaging behavior, such as landing on the ankle instead of the foot, is tagged as a failure and receives no reward. The training wheels for this proof of concept are a simple change of the total mass of the robot

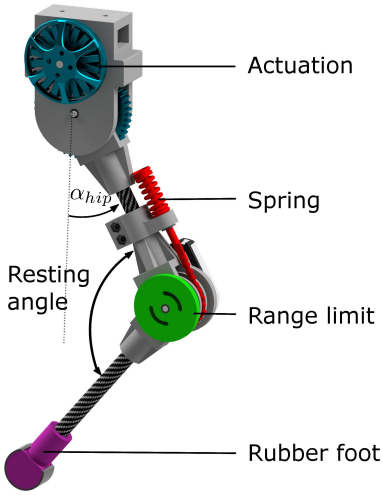


Fig. 3: The two-segment leg has a brushless DC motor at the hip, and a passive compliant ankle joint. The spring is mounted to a cam mechanism, and the joint itself is limited in extension range.

body: essentially we allow learning in a reduced gravity environment.
We choose these training wheels for two reasons: first, they should make it easier for the agent to learn the task. Second, they should also be easy to apply in practice, so the final behavior can be achieved with less engineering effort. Based on this we introduce our first criterion in designing training wheels: *how easy are training wheels to apply to a generic set of robots*. For example, the weight of batteries, computation or other payload can easily be offloaded during an initial training phase for most robot designs.
We choose a simple policy with a 2D parameter space: the hip actuator tracks an open-loop sinusoidal position trajectory as follows

$$\alpha_{hip} = \theta_0 + \theta_1 \sin(\omega t) \tag{1}$$

where α_{hip} is the angular position of the hip, ω is a hardcoded angular frequency while θ_0 and θ_1, offset and amplitude parameters respectively, form the parameter space of the policy. This simple policy parameterization serves two purposes: first, a low-dimensional deterministic policy is amenable to the simplest of learning schemes, and thus eliminates the ambiguity of whether the training wheels or the algorithm implementation are responsible for the change in performance. In the results presented we choose $\omega = 9$Hz, based on experience. Higher values achieve higher performance, but failures are also more violent and prone to damaging hardware. Since we also need to sample failure parameters to map out the landscape, we compromise between safety and performance.
We use stochastic gradient descent based on simple finite-difference methods [15]. More importantly, the low dimensionality allows us to empirically map out and inspect the landscape of the learning problem as a 3D surface as seen in Fig. 4. This allows us to compare the landscapes with and without training wheels in detail, and show the change in

learning performance across each landscape.

A. Hardware Details

Each DoF of the boom and leg is measured with a rotary encoder (CUI ATM102-V). The boom arm has a length of $1.5[m]$ from pivot to the leg, and is counterweighted to completely offset its own mass without the leg. The ankle joint of the leg (Fig. 3) is mechanically limited to 130° in one direction, and has a spring with a stiffness of 6 $[\frac{N}{mm}]$ attached to a cam mechanism with a radius of 15 $[mm]$. This spring is slightly preloaded such that it always returns to the resting angle of 130°. The upper and lower leg segments measure 110 $[mm]$ and 136 $[mm]$ respectively, and the virtual leg length from hip to foot is 223 $[mm]$ at rest. The hip is actuated with a brushless outrunner motor (T-motor MN-4006) with a 1:5 gearbox. The motor control board (Texas Instruments TMS320F28069M with DRV8305EVM booster packs) uses field-oriented control for direct torque control of the motor. A Xenomai real-time linux operating system handles high-level control. Electric power and computational power are both off-loaded via tether. A representative mass is directly attached on the boom just behind the leg. With the entire payload, the robot has a body weight of 600 $[g]$. For our two training wheel environments the representative mass is replaced with an intermediate mass or completely removed. This results in a body weight of 505 $[g]$ (0.84 g_0) and 415 $[g]$ (0.69 g_0) respectively.

III. RESULTS

We test three environments: the robot with full payload and two environments with training wheels which reduce the weight to 0.69 g_0 and 0.84 g_0, where g_0 is the weight of the robot in the original environment. We will refer to these two environments with training wheels as the *beginner environment* and the *intermediate environment* respectively. We map out the entire reward landscape for each environment by sampling and then interpolating the parameter space (Fig. 4). The parameter space is limited to $\theta_0 = [0\ 40]^\circ$ and $\theta_1 = [10\ 45]^\circ$. Parameters outside this range are either unreachable due to mechanical hard-stops, or in the zero-reward region for all environments, and cropped for clarity. All three landscapes have a mountain-like shape emerging out of a flat surface. While not quite convex, the landscapes each have a prominent peak, making them amenable to stochastic gradient descent. Also present in all three landscapes is a cliff: a sudden sharp drop from high to zero reward. This is found in the upper right quadrant of the parameter space and can be recognized in Fig. 4 by the dense contour lines. This cliff represents the border between parameters which exhibit stable high performance and unstable parameters. In practice it is both difficult as well as dangerous to learn from beyond this cliff: policies with high-amplitude tend to crash violently and damage the hardware. It is interesting to note that the orientation of the cliff does change in each environment, though its proximity to the peak does not.

A. Salient Gradient Sets

We are interested in the region that achieves non-zero reward which we will refer to as a *salient gradient set* (SCS), delimited in the figures by the thicker, outermost contour. This is the set of parameters a learning agent needs to sample from in order to learn. The second criterion for choosing effective training wheels is *how much the training wheels increase the size of the SCS*. Indeed the SCS of the beginner environment covers 46% of the total parameter space, compared to 25% of the intermediate environment and 20% in the original environment. This increase in size is important both for gradient-based and gradient-free methods. With stochastic gradient descent, for example, the gradient is estimated by local sampling. This means the agent must start inside, or at least within local sampling distance of the SCS in order to estimate a gradient. Increasing the size of the SCS directly increases the basin of attraction of the learning system. Other exploration approaches such as eps-greedy can sample from the SCS despite being initialized well away of the SCS. In this case, increasing the size of the SCS increases the probability of sampling from it, thus improving rate of convergence.

B. Funneling Sets

In Randløv's simulated work [12], the training-wheels environment converges gradually to the original environment. In practice, it is often difficult to implement a gradual mechanical change: in many cases it is desirable to have training wheels that are either on or off, or at least require only a few stages. This brings up an important requirement for training wheels: *successive environments must funnel into each other*. In Fig. 5, the peak of the beginner environment, located at $[22.5\ 32.6]^\circ$, lies only barely within the SCS of the original environment. In general, there are no guarantees the peak of the training-wheels will be contained in the SCS of the original problem. If it isn't, or in our case if the policy has not fully converged, there is a good chance that when the training wheels are removed, the policy is still too far from the next SCS to effectively sample from it. This issue is solved by having an intermediate environment, whose SCS contains a large area around the peak of the beginner environment. In other words, each training-wheel environment should easily *funnel into* the SCS of the next environment to be effective. This is particularly important when local exploration strategy is used, and is conceptually similar to designing controller funnels [16][17].

There is a second consideration that should be kept in mind: while the peak of an earlier environment *must*[2] be contained in the SCS of the successive environment, the reverse is not true. This means switching *back* to an earlier training environment must be done only cautiously, especially in hardware. A simple workaround is to keep a memory of previous policies, and switch back to known stable policies when switching back to previous training environments.

[2]This condition is necessary when exploration is strictly local, and can be relaxed otherwise.

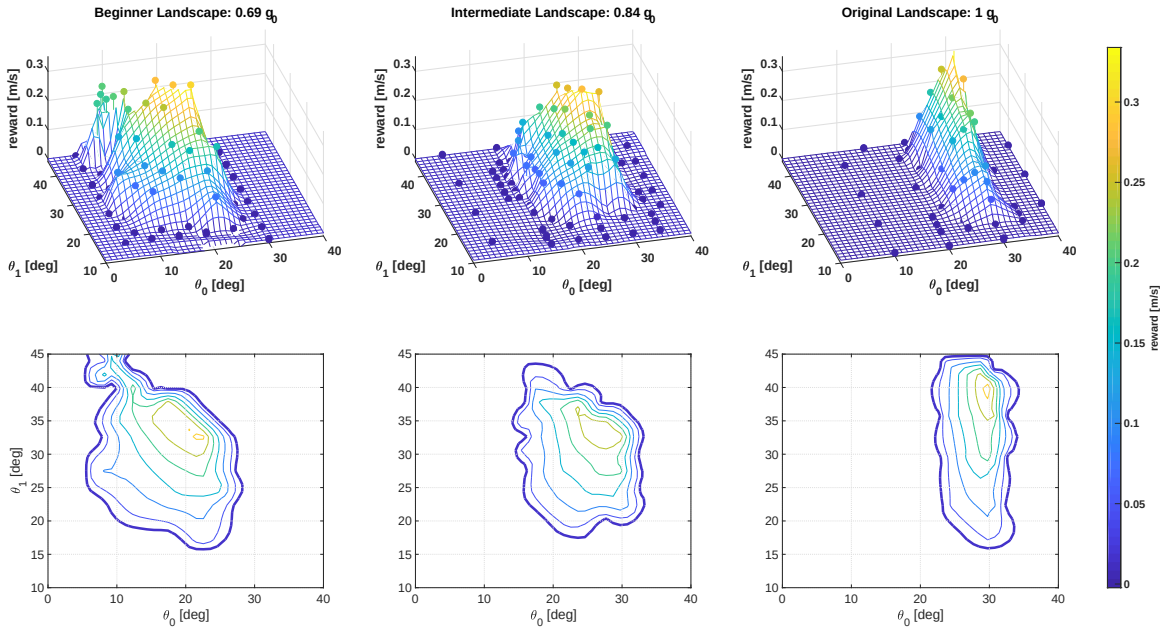


Fig. 4: The landscapes of the beginner, intermediate and original environments are visualized here. The upper row shows the sampled points (circles) and the resulting interpolated mesh, slightly offset for visibility. The more gradual climb in the lower gravity environments is visible. The contour maps in the second column more clearly show the change in shape of the 'reward mountain', the shape of of the cliff and most importantly, the size of the basin of attraction for the learning system. The outer contour showing the set of parameters which can provide a gradient is outlined with a thicker line. If the learning agent only samples outside this set, it will not be able to accurately estimate a gradient.

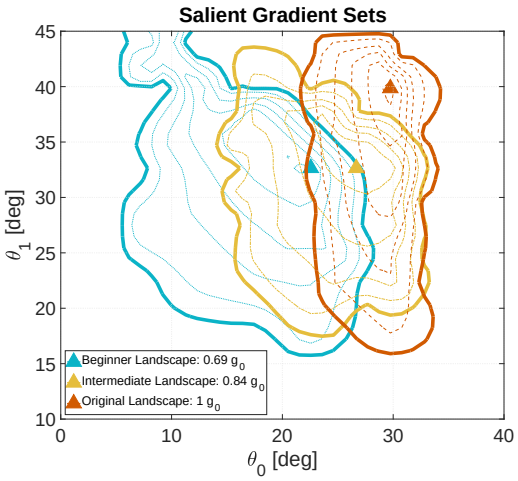
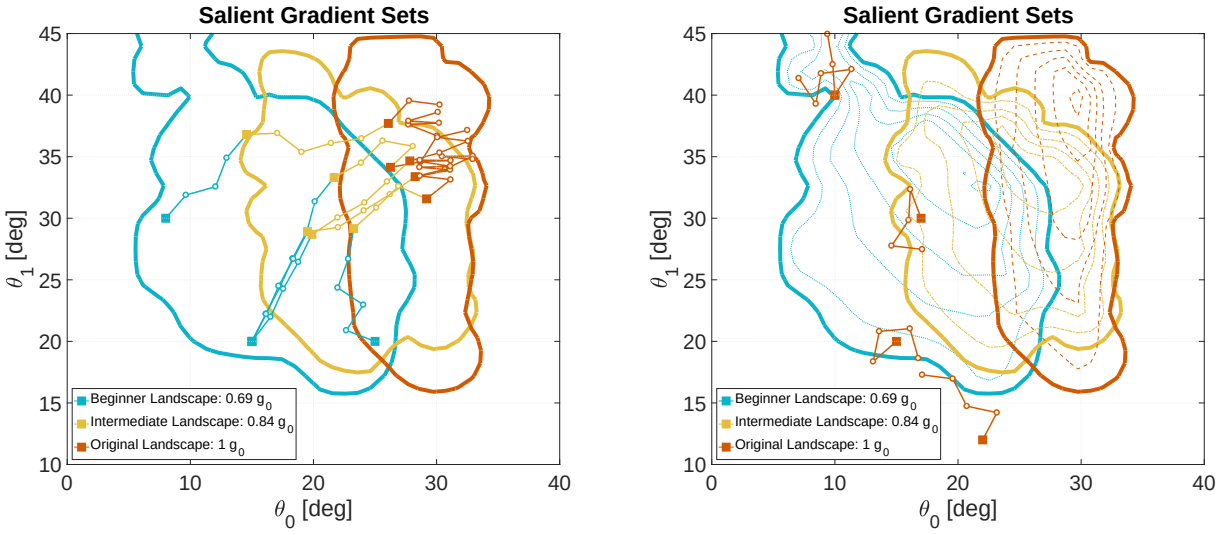


Fig. 5: The salient gradient set (SCS) for each environment is mapped out with contour lines and the peak of each set marked by a triangle. The location of the peak of one training environment with respect to the SCS of the successive environment is very important. To be effective, the training wheels must guide the current policy towards parameters that will sample from the salient gradient set of the next landscape with higher probability.

C. LEARNING ACROSS LANDSCAPES

As a proof of concept we use offline stochastic gradient descent with finite-differences, with parameter perturbations ranging between 0.5° and 2°. Gradient estimates tend to be more robust to noise with larger perturbations, especially where the gradient is very shallow such as at the edges of the SCS. On the other hand, they become unstable closer to the peak and especially when close to the cliff. We also choose a constant, relatively large learning rate of 2.5. Again, larger steps have the risk of overshooting and stepping over the cliff, but otherwise perform well. In both cases a cleverer, variable choice of these parameters would help the learning process, but is not relevant to the training-wheels concept and is kept constant.

Several typical learning sessions are shown in Fig. 6a, with the most successful reaching a speed of 0.35 $[\frac{m}{s}]$. We also purposefully initialize several trials outside the SCS of the original environment, and as expected we observe meandering paths. Examples of agents initialized with parameters outside their respective SCS are shown in Fig. 6b. As expected, without sufficient gradient information the agents will simply take steps in random direction. While this random exploration has a non zero probability of entering the salient gradient set and therefore converging, it can take a large amount of iterations, especially when starting at some distance from the set and with smaller learning step sizes. Especially when learning directly in robot hardware, it is critical to reduce the number of trials necessary.



(a) Typical successful learning paths

(b) Unsuccessful learning attempted with the original landscape, marked in red.

Fig. 6: Several typical learning paths are shown here: above, successful learning trials progress from the beginner to an intermediate and finally to the original environment. Contour lines are not shown for visual clarity. One of these learning trials typically took around 10 minutes. Below are trials initialized directly with the original environment but outside its salient gradient set. These take learning steps in random directions without improving.

IV. CONCLUSIONS AND OUTLOOK

We build on the concept of training wheels, temporary mechanical modifications of the system, to shape the learning landscape [12]. We apply it to learning open-loop legged locomotion in a constrained test stand, as a simple, low-dimensional problem that is unstable, underactuated and features impacts. We propose three criteria to designing effective training wheels in practice.

1) Ease of application to a generic set of robots
2) Increase in probability of sampling from the salient gradient set
3) Ease of funneling from a training environment to the successive environment

Since reducing the engineering effort is a main attraction for applying learning to robotics, it is important that training wheels are easy to implement and apply. As an example, we surmise that adding damping to the joints, or to the floating base, of a robot would help stabilize the system and greatly help learning by improving stability [18][19], at a moderate cost to performance and efficiency. Implementing mechanical damping on small joints is however much more difficult than simply temporarily offloading some of the payload, and would require a custom design for each new robot. This partially defeats the purpose of reducing the engineering effort. While we plan to explore solutions to this in future work, there is a lot of merit in solutions as effective yet simple as reducing the payload.

The second criterion is the main qualifier for the effectiveness of the training wheels in shaping the learning landscape. To be more precise, increasing the probability of sampling from the salient gradient set is what makes a training wheel environment easier to learn in. The actual size of the set in relation to the sample space is a good proxy; it is more generalizable to arbitrary exploration strategies, and makes it more intuitive to predict when designing training wheels. In most cases it will be impossible to map out the landscape by brute-force as we have done here. Good methods to approximate the SGS size, or directly estimate the sampling probability, need to be developed to more systematically evaluate potential training wheels.

The final criterion is particularly relevant when local exploration strategies are used. As this strategy is common in robotics, we feel it is an important criterion to include. Training wheels that can be continuously tuned out, until the dynamics converge back to the original environment, would be guaranteed to satisfy this criterion [12]. However the implementation of such training wheels generally goes against the first criteria, and a trade-off will have to be made. To be effective, training wheels will require a strategy to transition from one landscape to the next. In practice, it is helpful to regularly estimate the local gradient of the next landscape before each transition. Ideally, we would want to design a sequence of training wheels that funnel into each other, similar in concept to [16][20].

In this work, when to switch between environments was

chosen heuristically. With the actual landscape maps available for reference, we can be very confident that the funnel overlap between environments is large and we do not need to completely converge on one environment before switching to the next. For future work, it will be interesting to find a more general rule for switching environments. Since the number of trials needed to converge is particularly important when learning in hardware, an optimal switching policy to learn with the fewest iterations would be particularly useful.
Although we have presented these landscape shaping results in the context of reinforcement learning, the challenge of traversing a landscape in parameter-space is inherent to optimization problems as a whole. In particular, the concepts we develop should be useful for applying derivative-free optimization in hardware [21] as well.

ACKNOWLEDGMENTS

We would like to thank Özge Drama, Felix Grimminger and Julian Viereck for fruitful discussions, advice and help along the way. We also thank Felix Widmaier who wrote the code to run and interface with the motor control boards. We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting the academic development of Felix Ruppert. This work was partially supported by an MPI Grassroots grant provided to Ludovic Righetti, Felix Grimminger and Alexander Spröwitz in 2017.

REFERENCES

[1] X. B. Peng, G. Berseth, and M. Van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning", *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 81, 2016.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning", *arXiv preprint arXiv:1509.02971*, 2015.

[3] P. Abbeel, A. Coates, M. Quigley, *et al.*, "An application of reinforcement learning to aerobatic helicopter flight", in *Advances in neural information processing systems*, 2007, pp. 1–8.

[4] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients", *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[5] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[6] R. Tedrake, T. W. Zhang, and H. S. Seung, "Learning to walk in 20 minutes", in *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University New Haven (CT), vol. 95585, 2005, pp. 1939–1412.

[7] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion", in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, IEEE, vol. 3, 2004, pp. 2619–2624.

[8] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[9] N. Chentanez, A. G. Barto, and S. P. Singh, "Intrinsically motivated reinforcement learning", in *Advances in neural information processing systems*, 2005, pp. 1281–1288.

[10] V. Gullapalli and A. G. Barto, "Shaping as a method for accelerating reinforcement learning", in *Intelligent Control, 1992., Proceedings of the 1992 IEEE International Symposium on*, IEEE, 1992, pp. 554–559.

[11] T. McGeer, "Passive dynamic walking", *The International Journal of Robotic Research*, vol. 9, no. 2, pp. 62–82, 1990.

[12] J. Randløv, "Shaping in reinforcement learning by changing the physics of the problem.", in *ICML*, 2000, pp. 767–774.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 1. MIT press Cambridge, 1998, vol. 1.

[14] J. Rummel and A. Seyfarth, "Stable running with segmented legs", *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 919–934, 2008.

[15] J. Peters and S. Schaal, "Policy gradient methods for robotics", in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, 2006, pp. 2219–2225.

[16] Q. Cao, A. T. Van Rijn, and I. Poulakakis, "On the control of gait transitions in quadrupedal running", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 5136–5141.

[17] R. Tedrake, I. R. Manchester, M. Tobenkin, *et al.*, "Lqr-trees: Feedback motion planning via sums-of-squares verification", *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[18] J. E. Colgate and J. M. Brown, "Factors affecting the z-width of a haptic display", in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, IEEE, 1994, pp. 3205–3210.

[19] M. Calisti, F. Corucci, A. Arienti, *et al.*, "Dynamics of underwater legged locomotion: Modeling and experiments on an octopus-inspired robot", *Bioinspiration & biomimetics*, vol. 10, no. 4, p. 046 012, 2015.

[20] A. Karpathy and M. Van De Panne, "Curriculum learning for motor skills", *Advances in Artificial Intelligence*, pp. 325–330, 2012.

[21] A. Spröwitz, R. Moeckel, J. Maye, *et al.*, "Learning to move in modular robots using central pattern generators and online optimization", *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 423–443, 2008.

Beyond Basins of Attraction: Quantifying Robustness of Natural Dynamics

Steve Heim, Alexander Spröwitz
Dynamic Locomotion Group
Max Planck Institute for Intelligent Systems, Germany

***Abstract*—Properly designing a system to exhibit favorable natural dynamics can greatly simplify designing or learning the control policy. However, it is still unclear what constitutes favorable natural dynamics and how to quantify its effect. Most studies of simple walking and running models have focused on the basins of attraction of passive limit-cycles and the notion of self-stability. We instead emphasize the importance of stepping beyond basins of attraction. We show an approach based on viability theory to quantify robust sets in state-action space. These sets are valid for the family of all robust control policies, which allows us to quantify the robustness inherent to the natural dynamics before designing the control policy or specifying a control objective. We illustrate our formulation using spring-mass models, simple low dimensional models of running systems. We then show an example application by optimizing robustness of a simulated planar monoped, using a gradient-free optimization scheme. Both case studies result in a nonlinear effective stiffness providing more robustness.**

I. Introduction

Animals are not only agile and efficient, but also remarkably adaptable and robust [1], [2], with arguably simple control and morphology [3]–[5]. Reproducing this performance in legged robots has been difficult. Most robots use sophisticated algorithms [6]–[9] which rely on accurate models and state-estimation at a substantial computational cost. This reliance tends to make model-based approaches brittle.

Recently, there have been attempts to combine these approaches with machine learning to improve robustness and adaptability [10]–[12]; however, it is notoriously difficult to apply learning directly in hardware. We are motivated by the question 'how should a legged robot be designed, such that it is easier to apply model-free learning directly in hardware?'. A key part of the answer is the *inherent robustness of the natural dynamics* of the system.

Indeed, designing a system with favorable natural dynamics can simplify the control problem [13]–[17] and enable quick learning directly in hardware [18], [19]. It is, however, still unclear how to quantify and evaluate the effects of design choices on the control problem, especially in terms of robustness and ease of designing or learning the control policy. After a robot is deployed successfully, it is difficult to distinguish what is due to the mechanical design, controller design, implementation, or other factors. Designers must instead rely on experience and intuition.

Many studies of natural dynamics focus on the concept of self-stability and the basins of attraction of passively stable limit-cycles [20]–[23] or open-loop stable limit-cycles [24]. In this study, we advocate the importance of stepping away from thinking in terms of limit-cycles and their basins of attraction. We present a formulation grounded in viability theory which allows us to quantify the inherent robustness of the natural dynamics, prior to specifying the control policy parametrization or control objective.

Contact: heim@is.mpg.de or heim.steve@gmail.com

A. Natural Dynamics and Spring Mass Models

Perhaps the clearest example of natural dynamics is Tad McGeer's passive dynamic walker [25]: this purely mechanical system with no sensors or actuators (and hence no control) exhibits passively stable limit-cycles for downhill walking. This idea has been extended in several robots, adding a little actuation and control to allow walking on level ground [26], [27] and to increase the basin of attraction of the passively stable limit-cycle. A key concept is to *exploit the natural dynamics*. The intuition behind this concept is that the control can be 'lazy': if a perturbation pushes the system out of the basin of attraction, the control should guide it back in. Once the state is inside the basin of attraction, the control can allow the system to naturally evolve to the attracting limit cycle.

Simulation studies of idealized walking models such as the rimless wheel [28] and compass walker [29] have provided more understanding of McGeer's empirical results. These models also have passively stable limit cycles albeit with rather small basins of attraction.

For running, we turn to a different idealized model, the spring-mass model. This simple model was initially developed by the biomechanics community to study running [30], where the spring abstracts the natural compliance of the muscle-tendon system in the leg. While the effective leg stiffness depends on many factors including muscle activation, it is modeled as a constant parameter, and thus the model has no control inputs. Thus, at the level of abstraction of the model, the natural dynamics seem passive even though the system may have active control embedded in it.

This simple model, also called a *template*, accurately predicts the overall behavior of many seemingly very different systems, called *anchors* [31]. Indeed, by proper parameter tuning, the spring-mass model can be used to accurately model diverse running systems, from humans [32] to cockroaches [33], bipedal [14] to hexapedal [34] robots.

B. Templates, Anchors, and Hierarchical Control

Spring-mass model templates are often used for understanding hierarchical control [31] since the template and

anchor division offers a natural split in hierarchy. A high-level control policy can be designed based on a template in a low-dimensional space, while a low-level control policy based on the anchor is designed in the high-dimensional space. Thus, as long as the low-level controller enforces a template-like behavior on the system, the high-level controller design can be greatly simplified [35]–[37].

In this hierarchical context, the term natural dynamics is always relative to the level of abstraction being considered. Indeed, to a high-level control policy, there is no distinction between which part of the system behavior is truly 'passive' and which has been influenced by the low-level controller[1].

The template and anchor approach to hierarchical control has been used to develop various discrete-time high-level controllers: for example, the spring stiffness or landing angle of attack might be chosen once per step, but the continuous-time dynamics in between are left 'passive' [38]–[41]. One result with this approach is that choosing an open-loop trajectory of landing angles of attack during flight can achieve deadbeat control without active control during stance [24], [42].

While these results are impressive, they generally suffer from the curse of dimensionality: they are only tractable on the low-dimensional template models. Therefore, the high-level control relies on the overall system behaving as a simpler, lower-dimensional system. This is usually achieved through a combination of appropriate mechanical design, and a low-level controller that exposes a simpler dynamical behavior to the high-level controller.

There are two common approaches to low-level controller design. On the one hand, a low-level control policy can enforce the dynamical behavior of a specific template model [43]–[46]. While this approach offers more rigorous guarantees on the behavior of the high-level system, it is also generally more difficult to implement in practice.

On the other hand, the low-level control policy can be designed to produce a lower-dimensional behavior without enforcing the specific template dynamics [47]–[50]. This approach requires further tuning of the high-level control policy, since it explicitly allows for a mismatch between the high-level model and the actual system behavior.

Robustness is a key indicator of how accurate a model needs to be, regardless of the approach taken: a policy that is robust will suffer less from model inaccuracies. Our main contribution is a means to quantify the robustness of the natural dynamics, prior to designing the high-level control policy, or even specifying its objective. We first illustrate the quantification on template models in a rigorous manner. We then show an example application using gradient-free optimization to find robust parameters of a low-level controller, without enforcing a specific template model. We are thus able to quantify robustness without relying on low-dimensional template models.

C. Computation of Viability

Our quantification relies on the concept of viability: a state is said to be viable if there exists a set of control actions that keeps it inside the viability kernel for all time [51]. In other words, a state that starts outside the viability kernel will fail within a finite time, regardless of the control actions applied.

There has been much interest recently in computing viable sets and its dual, back-reachable sets [52], for safe control verification and design [53]–[56], and more recently safe learning of control [57], [58]. Our contribution complements prior work by using a viability formulation to quantify robustness of the system design prior to control policy design.

Viability-based approaches share a common challenge: computing viability kernels relies on gridding the search space, making the general case intractable [53], [59].

For particular classes of systems, more efficient algorithms have been developed to find either inner or outer approximations of viable sets, which can generally be scaled to 6-10 dimensions [59]. Thus, it is often beneficial to use approximations that fit these classes and dimension restrictions. Computation of viable sets is then performed on the low-dimensional approximation, which can be tracked using a hierarchical control strategy [60], [61].

This matches well with the existing template and anchor paradigm commonly used in legged robotics. We will show an example application, in which we optimize the parameters of the low-level control policy to exhibit robust natural dynamics to a high-level control policy.

D. Notes on Terminology

We use terminology common to the reinforcement learning community, such as actions instead of control inputs and control policies instead of controllers. We will speak of control policies *sampling* an action, or the system sampling a state-action pair, to indicate the policy can be stochastic.

Much of the mathematics in the paper revolves around sets in different spaces. Capital letters such as S denote spaces (in this case state space). Capital letters with a subscript such as S_F denote a set in the corresponding space, the meaning of the subscript being explained in the text (in this case the set of failure states).

E. Structure

In Section II we cover the details of the two spring-mass models we examine, their dynamics, and a typical bifurcation diagram for the SLIP model.

In Section III we compute the viability kernel as well as the transition map in state-action space. We illustrate how this encompasses the bifurcation diagram, and why bifurcation diagrams are limiting once we introduce control.

In Section IV we introduce our definitions of robustness, and how to use this to evaluate two different designs of leg compliance prior to designing a control policy.

In Section V we show an example application, in which the quantification developed is used as the fitness function to perform gradient-free optimization of a simulated planar monopedal robot.

In Section VI we summarize the key contributions of the paper, open questions, and our outlook.

[1]This is equivalent to the split between agent and environment in reinforcement learning.

II. Spring-Mass Models

We use two well-studied spring-mass models to illustrate our concepts: the spring-loaded inverted pendulum (SLIP) model and a nonlinear spring mass (NSLIP) model as first studied by Rummel and Seyfarth [23] (see Fig. 1). Both models have hybrid dynamics with the governing equations of motion switching between flight and stance phases.

During flight phase, the body follows a ballistic trajectory, whereas during the stance phase it follows a spring-mass motion, which depends on the modeled spring. The details of the equations of motion have been derived in [23], [30], and can be found in the appendix. For convenient comparison, we use the same parameters as in [23], which are similar to human averages. In this work, we consider only deterministic dynamics.

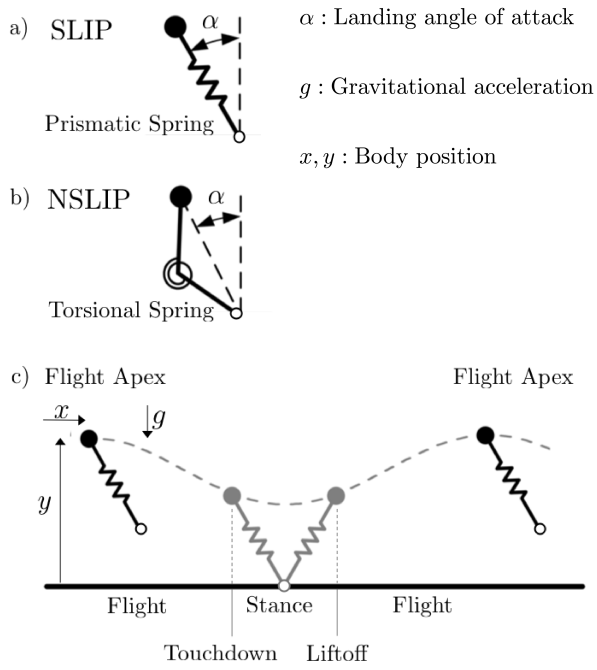


Figure 1: We focus on two spring-mass models: a) the spring-loaded inverted (SLIP) model with a linear prismatic spring, and b) a segmented leg model, with a linear torsional spring, which we will refer to as a nonlinear spring-mass (NSLIP) model. c) shows a qualitative trajectory over one cycle, starting and terminating with a flight apex event.

A. Discrete Analysis via Poincaré Sections

The continuous motion of the point-mass body is fully described in planar Cartesian coordinates by the state vector $[x, y, \dot{x}, \dot{y}]^\intercal$. We simplify analysis by only evaluating the state on a Poincaré section at flight apex, a common approach for cyclic motion. At flight apex, potential and kinetic energy are conveniently contained in the vertical position and forward velocity respectively. Thus, the continuous state vector of $[x, y, \dot{x}, \dot{y}]^\intercal$ can be reduced to $[y,\ \dot{x}]^\intercal$. Taking advantage of the constant energy constraint, we can further reduce the system to a single state, the normalized apex height s, which defines our state space:

$$s = \frac{E_{\text{pot}}}{E_{\text{pot}} + E_{\text{kin}}} = \frac{g\ y}{\frac{\dot{x}^2}{2} + g\ y}$$

$$\text{State Space: } s \in S = [0, 1]$$

where E_{pot} and E_{kin} are potential and kinetic energy, respectively, and g is the gravitational acceleration.

Starting from any state at apex s, we can numerically integrate the continuous time dynamics until the system either transitions to a second apex height or to a failure state. We thus obtain the Poincaré map, also called a transition map, for our discrete dynamics:

$$s_{k+1} = P(s_k, \alpha)$$

where the landing angle of attack α is a model parameter of interest. We use this as our control action in Section III.

We will consider as failures all states in which the body hits the ground with $y = 0$, as well as when the system reverses direction with $\dot{x} < 0$. More formally,

$$\text{Failure Set } S_F := \{s : y = 0 \text{ or } \dot{x} < 0\}$$

B. Bifurcation Diagram of the SLIP Model

A bifurcation diagram allows the study of the existence and stability of fix-points and limit-cycles, as a dependence of model parameters.

The bifurcation diagram of the SLIP model with respect to the angle of attack α is shown in Fig. 2. Similar bifurcation diagrams for spring-mass models can be found in [62], and bifurcation diagrams for spring stiffness can be found in [23], [38].

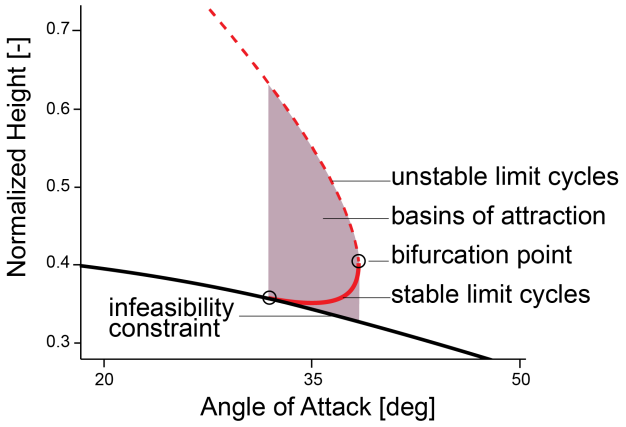


Figure 2: The bifurcation diagram of the passive SLIP model highlights the small range of parameters for which stable limit-cycles exist. The basins of attraction are bounded by infeasibility and unstable limit-cycles. Beyond these basins of attraction, however, is a lot of structure that can be exploited through control.

We only evaluate period-1 limit-cycles, that is when $s_{k+1} = s_k$, and do not consider orbits which require multiple iterations to return to periodicity. Stable limit-cycles are marked with

a solid red line and unstable limit-cycles with a dashed red line. The basins of attraction of the stable limit-cycles are highlighted by the shaded area.

These basins of attraction are bounded from below by an infeasibility constraint: below this line, the foot would begin underground. The unstable limit-cycles bound the basins of attraction from above: being perturbed onto an unstable limit-cycle will keep the system at that new state; beyond this threshold, it will diverge until the system fails.

Since either infeasibility or unstable limit-cycles bound the basins of attraction, many previous studies have been limited to identifying these bounds. The relevant range of parameters and states for studying basins of attraction tends to be narrow, as illustrated in Fig. 2. We will show in the next section that there is a lot of structure outside the basins of attraction of these passively stable limit-cycles. Once we allow parameters such as the angle of attack α to be actively chosen as a control decision, the relevant bounds are no longer the bounds of the basins of attraction, but those of failure and viability.

III. Natural Dynamics and Viable Control

We begin the section by introducing control, then evaluate the effect the natural dynamics have on the set of possible control policies. A key concept is the link between the viability kernel, a set within the state space, and the set of viable state-action pairs.

A. Control Policies and State-Action Space

We will now allow the system to choose the landing angle of attack α freely at each flight apex. This defines our action space A:

$$a = \alpha$$
$$\text{Action Space: } a \in A = [-180^\circ, 180^\circ]$$

where a is any action in A. In our figures we only show the relevant range, excluding the range which contains only failures or infeasible state-action pairs.

A control policy π is any function that maps a state to an action, $a = \pi(s)$. As such, a policy lives in the combined state and action spaces, which we term Q-space[2].

B. Transition Map

We compute high-resolution 800 by 800 grids of state-action pairs, as is commonly done for these types of problems [9], [24], [41], [56], [63]. We thus obtain a lookup table of the transition map $P(s_k, a_k)$, visualized in the state-action space Q in Fig. 3 for the SLIP model and in Fig. 4 for the NSLIP model.

To highlight the limit-cycles, we use a color-map centered around $s_k - s_{k+1} = 0$. The warm and cool colored regions correspond to state-action pairs that result in a higher or lower state, respectively. The gray regions are state-action pairs which result in a failure state $P(s_k, a_k) \in S_F$. The black region is composed of infeasible points in which the foot would start underground, and as such is not part of the Q-space.

[2]This term is chosen in reference to Q-learning in reinforcement learning.

We call the gray region the set of failing state-action pairs Q_F. Its complement, the colored region, is the non-failing set of state-action pairs Q_N. More formally,

$$Q_N \coloneqq \{(s_k, a_k) : P(s_k, a_k) \notin S_F\} \tag{1}$$

We denote the projection of Q_N onto the state space S as the set $S_N = \text{projs}(Q_N)$. Throughout the paper, we always use orthogonal projections, that is,

$$\text{projs}\,(s, a) = s \tag{2}$$

S_N is the set of controllable states, from which actions that avoid immediate failure can be selected. More formally,

$$S_N \coloneqq \{s_k : \exists\, a_k \text{ such that } P(s_k, a_k) \notin S_F\}.$$

The upper bound between Q_N and Q_F are state-action pairs that convert all kinetic energy into potential energy in one step, resulting in a state of $s = 1$. In other words, these are the equivalent of 1-step capture points [60]. The lower bound is a boundary to falling, meaning that the point-mass hits the ground without reaching a second flight apex.

C. Viable Sets

A viability kernel is the set of all states for which there is at least one time-evolution of the system which remains in the set for all time [51]. Since all state-action pairs $(s, a) \in Q_N$ result in at least a second step, all $s \in S_N$ have at least a one failure-preventing action available. However, it is possible for a non-failing state-action pair to reach a state from which all solutions eventually reach a failed state, as was examined in [64]. In other words, there can be states from which immediate failure can be avoided, but from which the system will fail within some finite time. Thus, the viability kernel, which we will call S_V, is a subset of S_N and the set of viable state-action pairs Q_V is a subset of Q_N.

We can compute the discretized set of viable state-action pairs Q_V and its projection S_V iteratively, as in Algorithm 1. In this process, we begin with an estimated $Q_V = Q_N$ and $S_V = \text{projs}(Q_V)$. Then we alternate trimming both estimates of Q_V and S_V: first, we check if any state action pairs (s, a) in the estimated Q_V maps to a state outside of S_V and exclude these from Q_V. Then we update the estimate of S_V as the projection of the new Q_V estimate and repeat. If the projection does not change, each state in S_V has an action available that maps back into itself and the algorithm terminates.

Algorithm 1 Compute Viable Sets

procedure VIABLE SETS(P, Q_N)
  $Q_V \leftarrow Q_N$
  $S_V \leftarrow \{\,\}$
  while $S_V \neq \text{projs}(Q_V)$ **do**
    $S_V \leftarrow \text{projs}(Q_V)$
    for all $s_{k+1} = P(s_k, a_k), (s_k, a_k) \in Q_V$ **do**
      if $s_{k+1} \notin S_V$ **then**
        Remove (s_k, a_k) from Q_V
  return Q_V, S_V

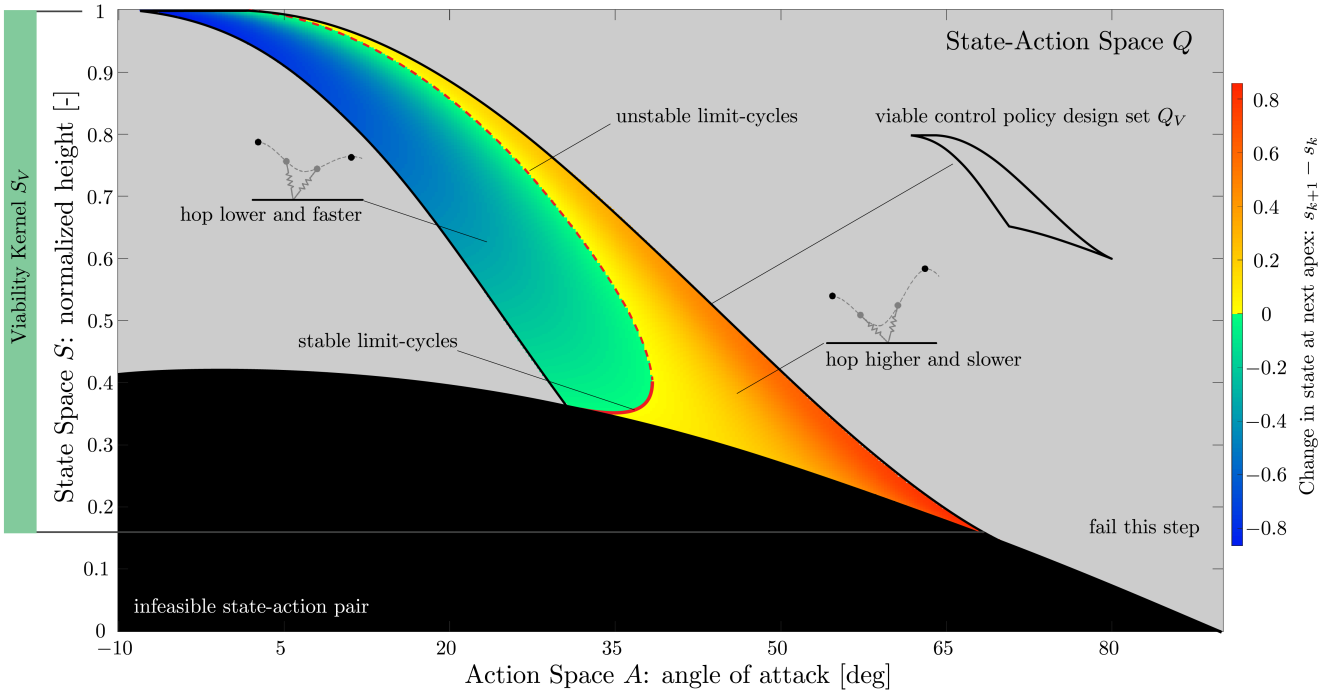


Figure 3: The lookup table of the SLIP model's transition map shows possible combinations of state (height at apex) and action (landing angle of attack), and their transition to either a second apex or a failure. State-action pairs in the gray region result in failure. State-actions in the warm and cool colored regions result in hopping higher and lower respectively, with the color indicating the change in state (vertical axis) at the next apex. Also marked are passively stable (solid red) and unstable (dashed red) limit-cycles, where the state does not change.

For the models we examine, Q_V is equal or almost equal to Q_N except in unusual corner cases.

We can now compare the resulting Q_V and S_V for the SLIP and the NSLIP models (Fig. 5). Although the set of viable states S_V is the same in both models, the set of viable state-action pairs Q_V is much larger for the NSLIP model. This suggests unexplored benefits of nonlinear leg compliance.

D. Family of Viable Control Policies

A control policy $\pi(s)$ must sample from Q_N with non-zero probability; otherwise, it will always fail in a single step. All meaningful policies must sample from Q_V with non-zero probability, or it will always fail in finite time. In order to avoid failure from every viable state for all time, a policy must sample *exclusively* from Q_V, which we call the viable policy design space. We call the set of all such policies the *family of viable control policies*. More formally, if the set Q_V is non-empty, we also have a non-empty set of viable policies Π_V, where

$$\forall s_k \in S_V \;\; \exists \;\; \pi(s_k) \in \Pi_V, \;\; a_k = \pi(s_k) : \\ (s_k, a_k) \in Q_V \text{ and } P(s_k, a_k) \in S_V \;\; \forall k$$

The shape of Q_V in the dimensions of S and A poses different constraints on the control policies $\pi(s) \in \Pi_V$ that we can design. The projection of Q_V onto the dimensions of state space S is the viability kernel S_V itself.

The volume of Q_V in the dimensions of action space A, on the other hand, allows more flexibility in designing a viable control policy since more viable actions are available to choose from.

Imagine for example a set Q_V defined by a single line[3] covering all of S, a surjective function $f(s)$. While the viability kernel $S_V = S$ is maximal, there is exactly one deterministic control policy $\pi(s) = f(s)$ which remains viable. This can make the control policy not only difficult to design or learn, but also very sensitive to uncertainty, as we will discuss in the next section.

IV. Robust Natural Dynamics

We define robustness as the ability of a system to avoid failure in the face of uncertainty. A key objective of this work is to evaluate the robustness inherent to the natural dynamics: we care about the robustness resulting from the system design, before specifying the policy parameterization or even the control objective (such as converging to a specific limit cycle).

To this end, we focus on uncertainty in action-space, in other words, the effect of noise on the control policy output. We will use this as a basis to also examine robustness to perturbations in state-space for the family of all robust controllers. We briefly discuss the link of action noise to state-estimation noise. We do not consider model uncertainty, and leave this to future work.

[3] A hypersurface for arbitrary dimensional state-action space.

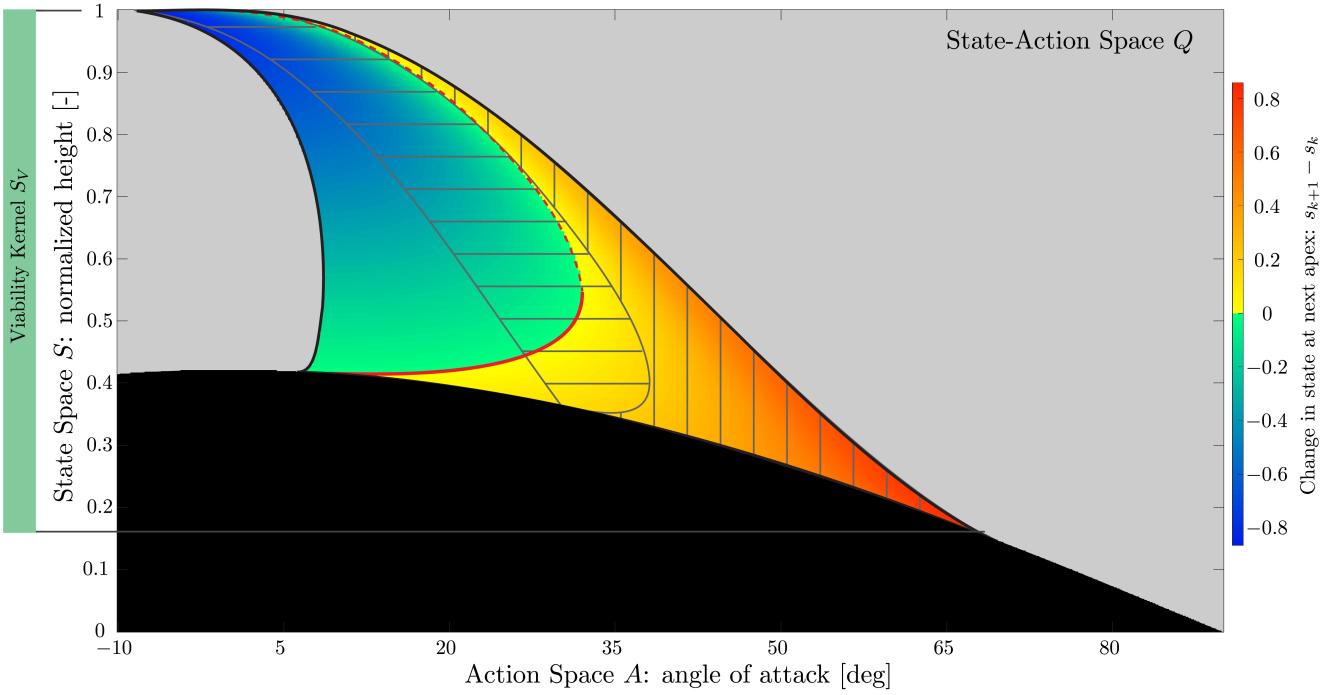


Figure 4: Although the viability kernel S_V remains the same for both models, the size of Q_V of the NSLIP is 36% larger. This allows for more flexibility and robustness in designing a control policy for the NSLIP model. For reference, the Q_V of the SLIP model with gray lines in horizontal and vertical for the cold and warm colored regions respectively.

A. *Computing Robust Sets*

Noise in the action space causes the system to sample a state-action pair with a different action than chosen by the policy:

$$a = \pi(s_k) + \eta_a \tag{3}$$

$$s_{k+1} = P(s_k, \pi(s_k) + \eta_a) \tag{4}$$

where η_a is some form of noise. A robust control policy needs to ensure that the chosen output never causes the system to fail despite this noise, for all time. More formally,

$$\begin{aligned} &\text{If } \pi(s_k) \in \Pi_R \text{ and } \eta_a \in H_a \\ &\text{Then } s_{k+1} = P(s_k, \pi(s_k) + \eta_a) \notin S_F \ \forall k \end{aligned} \tag{5}$$

where Π_R is the family of all robust control policies. For simplicity, we will consider noise sampled from a symmetrical bounded set $\eta_a \in H_a = [-\eta, \eta]$, where η is some finite scalar.

When considering unbounded noise (such as Gaussian noise), similar arguments hold in a probabilistic sense: instead of being able to guarantee that state-action pairs allow the system to never fail, we can only guarantee that it will not fail within a finite-time horizon with a certain probability.

The effect of action noise reduces the space available for controller design in two ways. First, the output of the control policy $\pi(s_k)$ must be sufficiently distant from failing state-action pairs, such that the added noise never causes an immediate failure. The second requirement is similar to that for viability: the system must always land in a state from which it can continue to sample robustly, for all time. More formally, we want that

$$\begin{aligned} &s_k \in S_R,\ \pi(s_k) \in \Pi_R,\ \eta_a \in H_a : \\ &P(s_k, \pi(s_k) + \eta_a) \in S_R \ \forall k \end{aligned} \tag{6}$$

We call Q_R the *robust control policy design set*. Similar to the relation between Π_V and Q_V, policies in the set Π_R must sample exclusively from Q_R in order to avoid failure for any state $s_k \in S_R$ where $S_R = \text{proj}_S(Q_R)$. Such sets are shown in Fig. 5 for various amounts of noise η. Each of these sets is computed with the iterative process in Algorithm 2. This is essentially the same as the algorithm for computing the viable set, while also considering additional possible transitions caused by noise. Note that, if the system dynamics have certain properties, only the worst-case noise needs to be considered [59]. Even without these properties, a worst-case only assumption is often sufficiently accurate in practice. Importantly, the computation of Q_R depends only on the set Q_V and thus the set of failure state S_F, the transition map P and the noise set H. It does not depend on the exact choice of policy $\pi(s_k)$, but is valid for the family of all robust control policies Π_R. In other words, we can evaluate the robustness inherent to the natural dynamics, before we design the control policy or define a control objective other than 'avoid failure'.

B. *Evaluating Robustness of Different Legs*

We compare the robustness of the SLIP and NSLIP models for varying amounts of noise, as shown in Fig. 5.

With the SLIP model, Q_R and S_R become empty sets for noise greater than $\pm 10.75°$, whereas in the NSLIP model the upper threshold is almost twice as large, at $\pm 20.00°$.

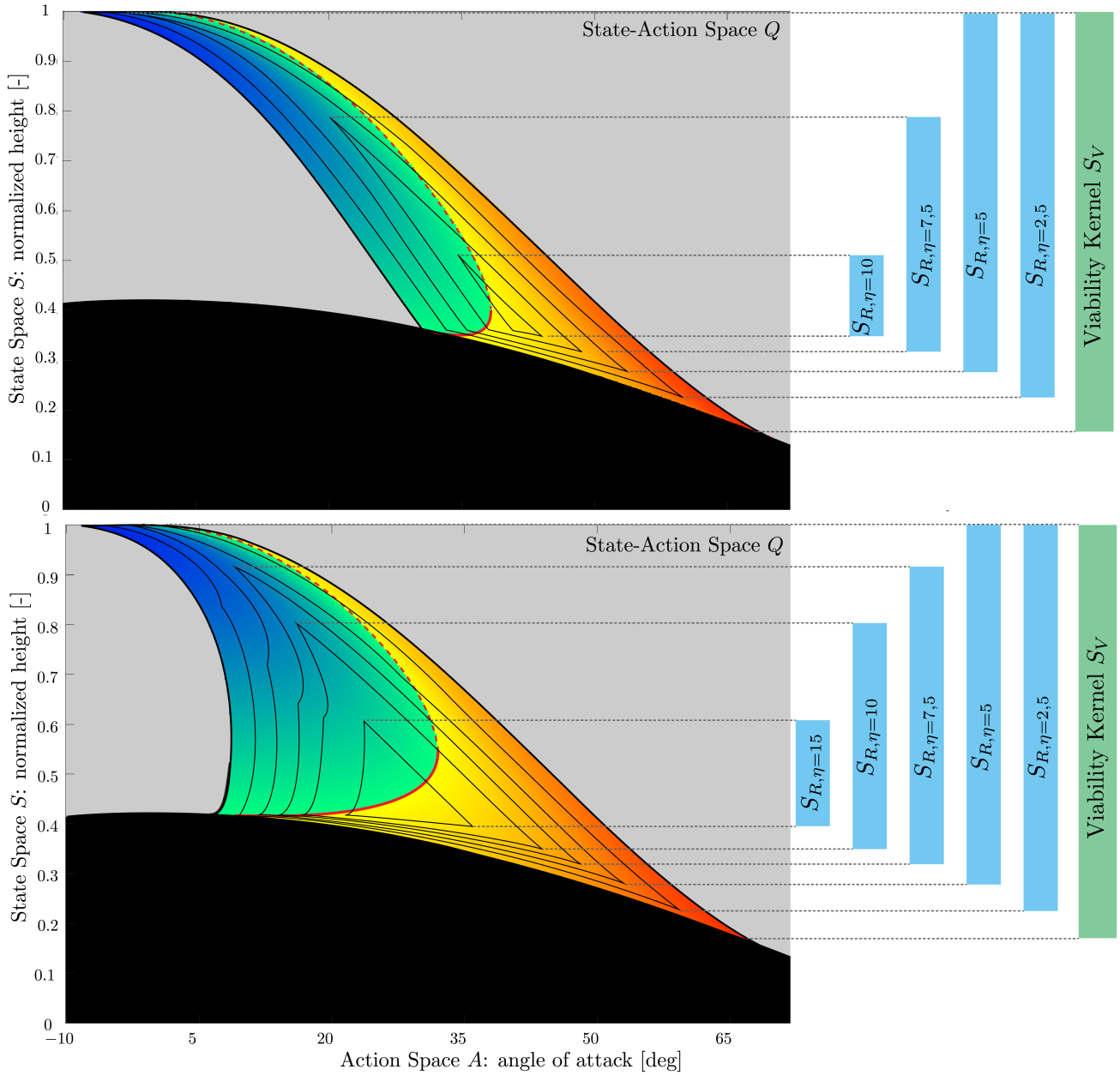


Figure 5: Robust sets for different amounts of noise are computed for the SLIP (top) and NSLIP (bottom). The NSLIP benefits from much larger robust sets Q_R for any amount of noise, which makes it easier to design or learn a robust control policy. Also, the set of robust states S_R are not only larger for the NSLIP, but remain relatively large even for rather imprecise control.

For any given amount of noise, the size of the set Q_R is also much greater for the NSLIP than for the regular SLIP model. The larger size of Q_R means there is more flexibility to fulfill robustness requirements while also designing a control policy around other criteria.

Furthermore, action noise is one of the most common methods of introducing exploration in learning, for example with Gaussian policies [65]. The amount of noise needs to be carefully balanced: more noise allows for more aggressive exploration, but it can also keep the agent from converging to the true optimum, as well as lead to unstable behaviors ending in failed states. This can be particularly troublesome for learning in hardware, requiring more samples as well as potentially damaging the robot. Robustness to action uncertainty allows for more aggressive and effective exploration during learning. This is particularly important for applying model-free learning directly in hardware.

C. Robustness to State Perturbations

The projection of the robust policy design set onto state-space, $S_R = \text{proj}_S(Q_R)$, is the set of robust states, from which any robust policy $\pi \in \Pi_R$ can always recover. Interestingly, with small amounts of noise up to $\eta < 5°$, S_R remains the same for both the SLIP and NSLIP models (see Fig. 6). For greater amounts of noise, it shrinks much more rapidly for the SLIP model.

Algorithm 2 Compute Robust Sets

procedure ROBUST SETS(P, Q_V, H)
  $Q_R \leftarrow Q_V$
  $S_R \leftarrow \{\}$
  while $S_R \neq \text{proj}_S(Q_R)$ **do**
    $S_R \leftarrow \text{proj}_S(Q_R)$
    for all $(s_k, a_k) \in Q_R$ **do**
      for all $\eta_a \in H_a$ **do**
        if $(s_k, a_k + \eta_a) \notin Q_R$ **then**
          Remove (s_k, a_k) from Q_R
          Break
        if $s_{k+1} = P(s_k, a_k + \eta_a) \notin S_R$ **then**
          Remove (s_k, a_k) from Q_R
          Break
  return Q_R, S_R

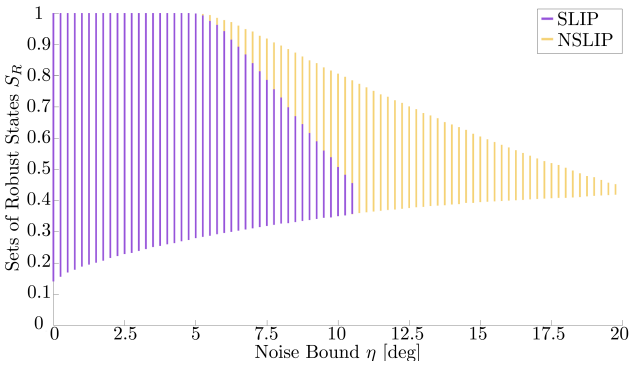


Figure 6: The size of the sets of robust sets S_R remains equal for the SLIP and NSLIP models for noise bounded to less than 5°. For greater amounts of noise, the sets shrink much more rapidly for the SLIP model.

The set S_R is particularly useful for choosing the specific control objective. For example, if we expect perturbations in state-space to have a symmetrical distribution, we would want to stabilize a limit-cycle near the center of S_R. On the other hand, if we expect a specific type of perturbation to occur more frequently, we can choose a limit cycle with a larger margin in that specific direction.

As a specific example, a well-studied state perturbation is a change of ground height between steps [1], [13], [24]. This type of perturbation involves a change in total energy: the forward velocity at apex remains the same, though the effective height (and thus potential energy) changes. We can compute S_R at different energy levels to then pick out operating points that remain robustly controllable across different energy levels, as shown in Fig. 7. Assuming symmetric distribution of perturbations, the control objectives should be chosen to maximize the distance from the edge of the viability kernel in each direction. For a given desired forward velocity, we can thus choose a total energy that centers the normalized height to perturbation along the vertical axis (constant energy perturbation) and along the forward velocity isolines (ground height change).

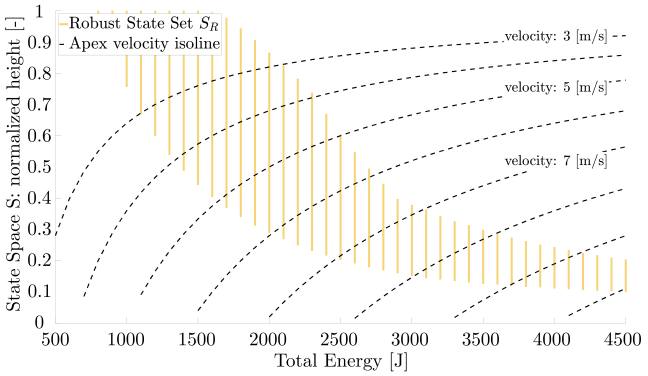


Figure 7: We show here the S_R for different amounts of total energy for the NSLIP model, with noise fixed at $\eta = 7.5°$. For a change in ground height, the system state travels along the forward velocity isolines (dashed black). For reference, the author runs recreationally at roughly $3.2\,[m/s]$, Eliud Kipchoge ran the Breaking2 marathon event at roughly $5.8\,[m/s]$ and Usain Bolt holds the 100 meter dash world record at roughly $10.8\,[m/s]$. The simulations shown in other graphs are all for the fixed energy level of 1'860 Joules.

D. Robustness to State Estimation Uncertainty

Sensory noise causes the control policy to sample an action based on a noisy estimate of the state:

$$a = \pi(s + \eta_s) \tag{7}$$

where η_s is the noise in state space. There is an equivalence between η_s and η_a: the action used deviates from what a control policy would determine under perfect conditions, whether this is due to noise in action space or state estimation. This equivalence can be directly calculated using eq. 4 and eq. 7:

$$\pi(s) + \eta_a = \pi(s + \eta_s)$$
$$\eta_a = \pi(s + \eta_s) - \pi(s)$$

If the control policy π is affine, the equivalence is trivially $\eta_a = \pi(\eta_s)$ and for bounded estimation noise η_s the equivalent action noise η_a is also bounded. Otherwise, we cannot guarantee bounds are available. Since this equivalence is dependent on the specific control policy, we do not investigate it further here. Suffice it to say, increasing robustness to action uncertainty can only improve robustness to state-estimation uncertainty as well.

E. Model Comparison

Previous studies of spring-mass models by Rummel and Seyfarth and others [23], [66], [67] have suggested that nonlinear effective leg compliance can improve stability. These studies focus on finding basins of attraction with a fixed parameter set. As such, they focus specifically on limit-cycle motion and only provide insight to robustness to state perturbations.

With their numerical studies, Rummel et al. show that, compared to a linear leg compliance, a nonlinear leg compliance has a broader range of parameters which exhibit passively stable limit-cycles. These limit-cycles also tend to have larger basins of attraction. However, at higher velocities, the model

with nonlinear spring stiffness no longer exhibits passively stable limit cycles, whereas with a linear spring this property is retained. These results suggested that nonlinear compliance is only beneficial at lower running speeds [23].

Using our formulation, we can evaluate robustness to state perturbations not only for an open-loop system but for any robust control policy. Our results confirm that, even with a maximally robust control policy, the set of robust states S_R shrinks at higher speeds (see Fig. 7), though not as drastically as the basins of attraction studied by Rummel et al.

V. Optimizing Natural Dynamics for Robustness

As an example application, we use our quantification to optimize the robustness of a simulated planar monoped with a 2-segment leg with a hierarchical control structure, shown in Fig. 8. The kinematic tree of the simulated system matches a robot testbed we currently use in our lab, though we have adjusted the parameters to be consistent with the models in the previous section. The system consists of three links: a floating-base free to move in the plane, but without rotation, and a two-link leg. Both hip and knee joints are actuated, resulting in an 8-dimensional state space and a 2-dimensional action space. Rigid impacts and ground-reaction forces are solved as described in [43], [68].

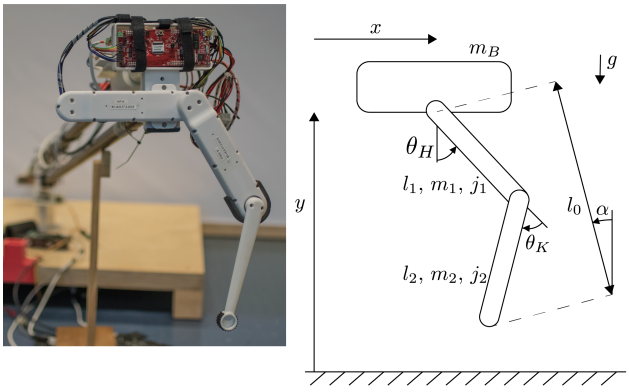


Figure 8: The simulated system is based on a hardware testbed, which is rigidly attached to a boom. Thus the floating base is limited to two degrees of freedom. Two additional degrees of freedom, the hip and knee joints, are both actuated. Thus the system has 4 position coordinates $q = [x, y, \theta_H, \theta_K]^\intercal$, an 8 dimensional state space $[q, \dot{q}]^\intercal$ and a 2 dimensional action space $[\tau_H, \tau_K]^\intercal$, where τ_H and τ_K are the hip and knee torques, respectively. The robot shown is designed by our colleague Felix Grimminger.

We use the volume of the robust set Q_V as the fitness function for a particle swarm optimization (PSO), a standard gradient-free optimization scheme. Thus, instead of requiring the low-level controller to enforce a specific template model, we improve its robustness in a general sense. The resulting natural dynamics allow for a high-level control policy to be implemented more reliably.

A. High-Level State-Action Space

The choice of the high-level state-action space is based on the spring-mass models and classic Raibert control [69], which share many similarities. The structure is shown in Fig. 9.

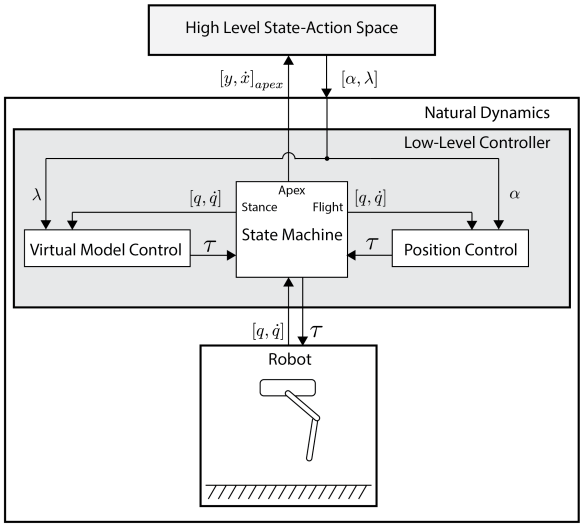


Figure 9: The high-level state-action space is composed of the height and forward velocity of the floating base at apex $[y, \dot{x}]^\intercal_{apex}$, the desired landing angle of attack α and the thrust factor λ. The natural dynamics considered are those relative to the high-level. These include both the rigid-body dynamics of the simulated robot as well as the embedded low-level controller.

The state is defined on the Poincaré section at flight apex, as introduced in Section II. Since the system is not energy-conservative, both the height and forward velocity of the floating base at apex must be considered, resulting in the state vector $[y, \dot{x}]^\intercal_{apex}$.

The action space is defined as a desired landing angle of attack α, constrained within 0 and 45°, and a thrust factor λ applied during stance, constrained within 1 and 2. This results in a 4-dimensional state-action space in the high-level, which is amenable to direct computation of a sufficiently dense grid.

Although our choice of the state-action space is largely motivated by Raibert control, we make no restrictions on the high-level control policy and do not decouple the states and actions.

B. Low-Level Controller

The low-level controller is a state-machine that switches between flight and stance.

During flight, a standard PD position controller tracks the desired landing angle of attack α dictated by the high-level control policy. The resting length of the virtual leg, l_0, is set as a constant parameter less than the maximum leg length to avoid reaching singularities. Thus α uniquely determines the desired foot position during flight. Since there are two possible joint configurations for each desired foot position, this orientation is also set as a constant parameter in the computation of the inverse kinematics. Thus α also uniquely determines the desired joint angles. During the first flight phase, from apex till touchdown, α is freely chosen as the action. For the second flight phase, from liftoff till the next apex, α is reset to the default position 0. Thus the initial leg configuration at each apex is expected to be the same.

During stance, we do not enforce the dynamics of a spring mass template model. Instead, compliant behavior is achieved

via virtual model control (VMC) [47], [49]. Torques are computed to mimic a relatively arbitrary leg compliance:

$$[\tau_H, \tau_K]^\intercal = \begin{cases} BJ_c^\intercal \, k_v \Delta l + K_j \Delta\theta & \text{if } \dot{y} < 0 \\ \lambda \left(BJ_c^\intercal \; k_v \Delta l + K_j \Delta\theta \right) & \text{otherwise} \end{cases} \quad (8)$$

where $[\tau_H, \tau_K]^\intercal$ are the hip and knee torques, B is the actuator selection matrix, J_c is the contact Jacobian, k_v is the stiffness coefficient of a virtual linear spring between hip and foot, Δl is the deflection of the virtual leg from rest, K is a symmetric linear matrix, and $\Delta\theta$ is the joint deflection of the leg from rest. The diagonal coefficients of K can be interpreted as virtual springs on the corresponding joints, while the off-diagonal coefficient serves as a mixing term. As long as K is positive-definite, K results in a nonlinear compliance with respect to the virtual leg deflection Δl. In similar fashion to classic Raibert control [69], additional thrust is triggered once the body reverses direction by amplifying joint torques by the thrust factor λ, as dictated by the high-level control policy.

We assume accurate tracking of α during flight phase, which is achieved through proper tuning of the PD gains. This is important to ensure well-behaved high-level dynamics for two reasons. First, to ensure that each high-level state-action pair results in a unique state at touchdown. Second, to ensure that the robot leg returns to the same resting configuration at each apex. In this manner, the leg masses can be lumped with the floating base to determine potential and kinetic energy, meaning that the high-level state $[y, \dot{x}]^\intercal_{apex}$ fully describes the system energy. The transition map P thus provides a unique map for each high-level state-action pair, and the viable sets S_V and Q_V can be directly computed in the high-level state-action space.

C. Optimization Setup

We use a standard PSO implementation based on [70]. The parameters optimized are the stiffness coefficients of the virtual leg in the low-level stance controller, $[k_v, k_{11}, k_{22}, k_{ij}]$, where k_{11} and k_{22} form the diagonal of the symmetric matrix K, and k_{ij} is the off-diagonal term.

As fitness function, we choose to maximize the hypervolume enclosed by the viable set Q_V in the high-level state-action space. For our systems, we have found that maximizing the hypervolume of Q_V and Q_R generally leads to the same results for reasonable amounts of noise. Each dimension of the state is normalized by heuristically determined bounds on maximum height and forward velocity, and the dimensions of the action space are bounded by their corresponding constraints. The hypervolume is calculated by summing and then normalizing the points inside the set. Thus, a fitness of 1 means that for any state, all actions are viable. A fitness of 0 means that for any state, all actions are outside the viable set.

For the results shown, 25 particles were initialized at random. Convergence tolerance on the fitness variance was set to 10^{-5}, which was reached after 12 iterations, taking roughly 3.5 hours on a 28-core desktop. During the optimization, we used a low-resolution grid with 160'000 points to speed up computation. Note that a lower resolution will result in a more conservative estimate of the sets, but not in mislabeled points in the set. The simulation parameters used are:

Mechanical Parameters			
gravitational acceleration	g :	9.81	$[m/s^2]$
body mass	m_B :	65	$[kg]$
upper leg length	l_1:	0.5	$[m]$
upper leg mass	m_1:	10	$[kg]$
upper leg inertia	j_1:	2	$[Kgm^2]$
lower leg length	l_2:	0.5	$[m]$
lower leg mass	m_2:	5	$[kg]$
lower leg inertia	j_2:	2	$[Kgm^2]$
Low Level Control Parameters			
leg resting length	l_0 :	0.85	$[m]$
saturation torque	τ_{max}:	2000	$[Nm]$
Hip joint PD gains	$[k_p, k_d]$:	$[500, 50]$	$[-]$
Knee joint PD gains	$[k_p, k_d]$:	$[500, 25]$	$[-]$

D. Optimization Results

We compare the robustness with a virtual leg compliance roughly matching that of the SLIP model, with stiffness coefficient $[k_v, k_{11}, k_{22}, k_{ij}] = [8, 0, 0, 0]\, 10^3$, versus one with the stiffness coefficients resulting from the optimization, $[k_v, k_{11}, k_{22}, k_{ij}] = [8.1, 5.0, 0.9, -0.5]\, 10^3$. The viability kernels S_V are visualized in Fig. 10. The intensity of the color-map indicates the portion of the action space which is viable for each point in state space. The red triangle marks an arbitrary operating point, $[y,\ \dot{x}]^\intercal = [1, 1]^\intercal$, and the action space for this state is shown in the image inset. In the action space, the action-pair leading to limit-cycle motion is also marked by a red triangle. To illustrate improved robustness, 50 actions are uniformly sampled around the operating point assuming bounded noise $\eta = [5^\circ, 0.1]^\intercal$ (orange circles) and an additional 50 with bounded noise between η and 2η (blue circles).

As in the comparison between the SLIP and NSLIP models in the previous section, the viability kernel S_V in state space remains nearly identical for both systems. The volume of the set of viable state-action pairs, however, increases from 0.08 to 0.23, over 2.8 times. The noisy sampling of actions around the operating point shows the decreased sensitivity to action noise with the optimized nonlinear compliance. In Fig. 10 we chose an arbitrary operating point for the sake of simplicity and fair comparison. In practice, an operating point can be chosen based on the robustness of that point in state-space. Conversely, instead of optimizing the overall robustness of the system, the fitness function can be weighted to bias robustness near a predetermined operating point.

VI. Conclusion and Outlook

We have presented a formulation for computing viable and robust sets in state-action space which allows the inherent robustness of a system to be quantified, prior to specifying the control policy parameterization or objective. Different system designs can thus be compared quantitatively.

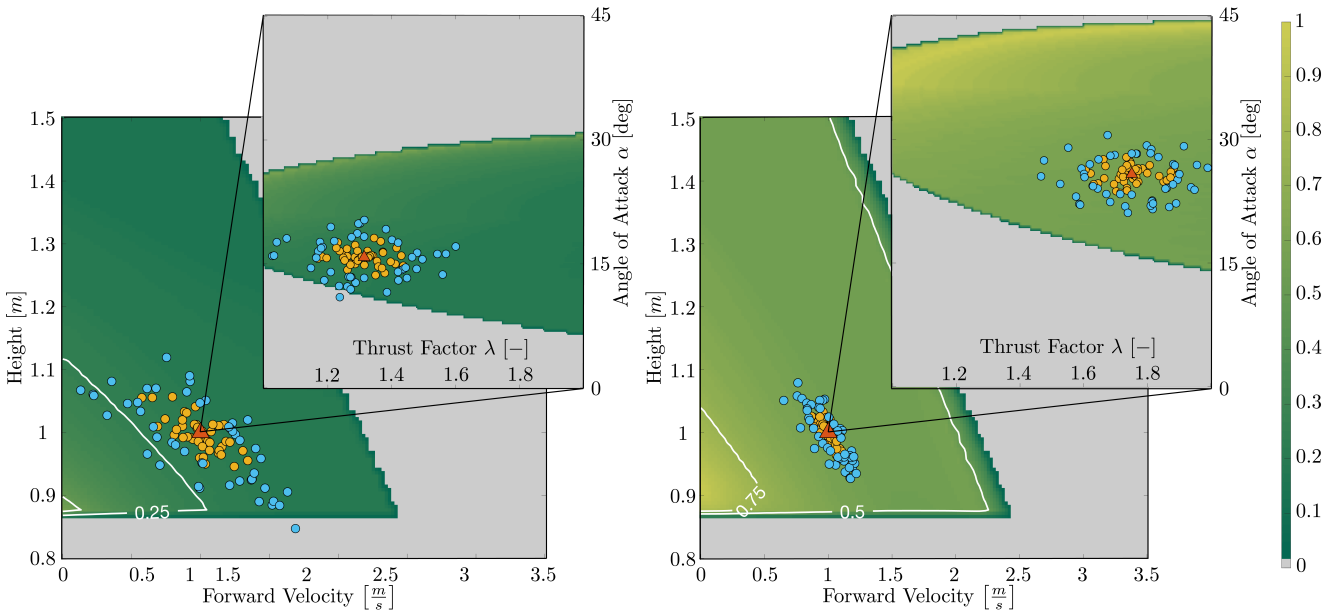


Figure 10: Shown are the viability kernels in the high-level space for the initial monoped (left) and after optimizing virtual compliance (right), along with the action-space for the operating point $[y,\ \dot{x}]^\mathsf{T} = [1,1]^\mathsf{T}$, shown in the image insets. A red triangle marks the state-action pair which leads to limit-cycle motion on the operating point, in both the state and action spaces. In the action space, the orange and blue circles mark actions randomly sampled around the operating point and with bounded noise $\eta = [5°, 0.1]^\mathsf{T}$, and between η and 2η, respectively. The states reached by these state-action pairs are marked with their respective colors in the state-space, which shows the much lower sensitivity experienced by the optimized monoped. The intensity of the color-map indicates for each point in state space, the portion of the action space which is viable. In the action space (image insets), the color-map indicates the intensity of the state that would be reached if that state-action pair were sampled.

We have illustrated this formulation on the spring-mass model, a low-dimensional system commonly used to synthesize control strategies for running robots. Furthermore, we have shown an example application using our quantification to perform gradient-free optimization. The system optimized is a simulated planar monoped with a two segment leg and a hierarchical control structure. The low-level controller parameters are optimized to improve robustness of the natural dynamics, as relative to the high-level state-action space.

An important advantage of this formulation is that the natural dynamics robustness can be optimized without enforcing the dynamics of a specific template model, which is often challenging and requires extensive tuning, developing accurate models as well as state estimation [44], [71], [72]. Instead, the inherent robustness will allow control policies designed on simple model abstractions to be leveraged despite inaccuracies.

To the best of our knowledge, prior work in viability theory focuses on evaluating robustness of a specific control policy, or on synthesizing control policies directly, and computation is limited to viability kernels in state space.

The notable exception is the work of Zaytsev et al. [56], which also computes viable sets in state-action space. Aside from the minor difference in studying walking instead of running models, Zaytsev et al. focus on the connection between controllability and viability. This is used to qualify how robust a given control policy is, how appropriate different templates may be for a given control task and given robot, and to motivate the statement that planning two steps ahead is sufficient. While we use the same state-action space formulation, we take a different approach to quantification by evaluating bounded noise in action space, which is more suitable for our motivating question: how to design natural dynamics that are easy to exploit? Indeed, we show why this is the only type of uncertainty which can be considered for the family of all robust control policies, without setting any assumptions on the control policy structure or objective. As such, we find our methods to be highly complementary, and applicable at different stages of robot design.

One of the main challenges with viability-based approaches is tractability [53], [59]. While we have shown how, in principle, a hierarchical control scheme reduces dimensionality, this approach alone is rarely sufficient in dealing with the curse of dimensionality on real systems. There is much recent progress on different scalable approaches to computing viable and backreachable sets (see Section I-C), and the specific choice will depend greatly on the properties of the system in question.

For running motion, characterized by nonlinear, non-smooth hybrid dynamics, we believe that, in addition to dimensionality reduction through hierarchical control, the use of heuristics such as computing ahead only two steps [56], are among the most promising tools to scaling this to real hardware.

We are also interested in using sampling-based approaches to make probabilistic estimates. There has been keen interest recently in applying machine learning techniques to tune control parameters directly in hardware [10], [73]–[75]. In these situations, safe exploration of the state-action space is particularly important. Active sampling to add samples close to the edge of the viable set would significantly increase sample-efficiency for estimating the sets, while at the same time allowing safe exploration, making this a logical next step.

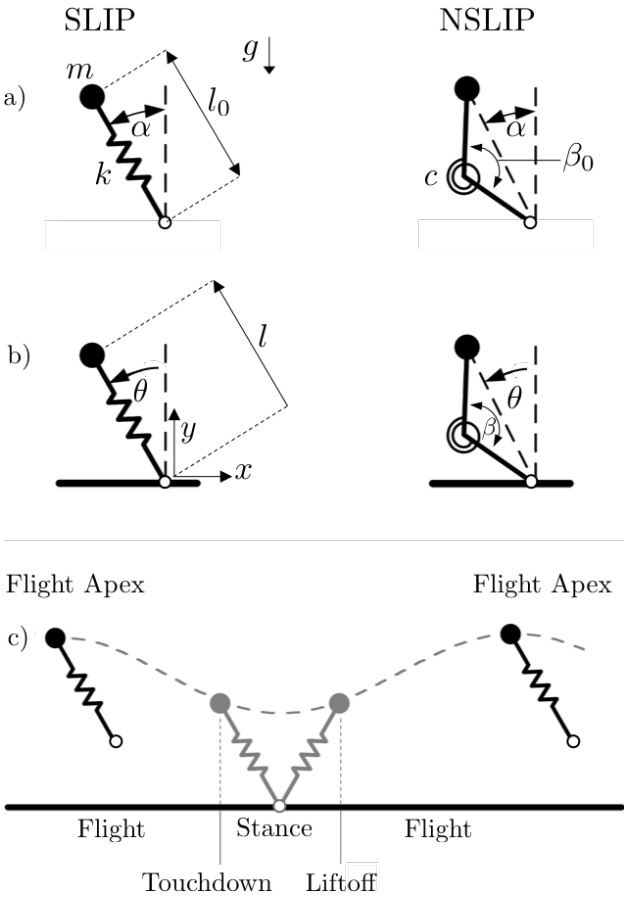


Figure 11: a) shows the parameters of the SLIP and NSLIP models. b) shows the states. The reference frame is reset to the foot position at each touchdown. c) shows a qualitative trajectory over a full cycle, with the relevant phases and events.

There is also potential for improvement in the definition of failures, the starting point of any viability approach. In this paper, we have used a very general and intuitive definition for failure (falling and direction reversal), however other definitions may be equivalent while offering earlier detection when computing viability kernels. Conservative definitions which lead to inner approximations may also be useful if they substantially speed up computation. It may also be possible to decouple the system dynamics, a common approach to simplifying control [15], [69], [76], and identify different failure conditions for each decoupled subsystem. This divide and conquer approach would also allow substantially higher dimensional systems to be tackled.

APPENDIX: SLIP AND NSLIP MODELS

The SLIP and NSLIP models are shown in Fig. 11. Integration between two apex events is split into three phases: a flight phase which terminates with a touchdown event, a stance phase which terminates with a liftoff event, and another flight phase which terminates with an apex event. The flight phase equations of motion are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \end{bmatrix}$$

where x and y are the body position and g is the gravitational acceleration. The stance phase equations of motion are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \frac{F_{leg}}{m} \begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix} - \begin{bmatrix} 0 \\ g \end{bmatrix}$$
$$\theta = \arctan 2\left(\frac{y}{x}\right) - \frac{\pi}{2}$$

where θ is the incident angle between the body and the foot (the rotation by $\frac{\pi}{2}$ serves to keep it consistent with the landing angle of attack) and F_{leg} is the force acting on the body due to the spring. In the SLIP model,

$$\text{SLIP: } F_{leg} = k(l_0 - l)$$
$$l = \sqrt{(x^2 + y^2)}$$

where k is the spring coefficient, l_0 is the spring resting length, and l is the leg length. In the NSLIP model,

$$\text{NSLIP: } F_{leg} = \frac{4lc(\beta_0 - \beta)}{l_0^2 \sin(\beta)}$$
$$\beta = \arccos\left(1 - \frac{2l^2}{l_0^2}\right)$$

where c is the torsional spring coefficient, β_0 is the spring resting angle and β is the knee angle. The three events are

$$\text{touchdown: } l = l_0$$
$$\text{liftoff: } \theta = \arctan 2\left(\frac{y}{x}\right) - \frac{\pi}{2}$$
$$\text{apex: } \dot{y} = 0$$

At each touchdown, the reference frame is reset to the foot position, which allows the equations of motion to be written more compactly. In the simulation, we also keep track of the foot position in an auxiliary variable.

For convenient comparison, we use the same parameters as in [23], which are similar to human averages:

gravitational acceleration	g :	9.81	$[m/s^2]$
body mass	m :	80	$[kg]$
prismatic spring resting length	l_0:	1	$[m]$
prismatic spring coefficient	k :	8200	$[N/m]$
torsional spring resting angle	β_0 :	170	$[^\circ]$
torsional spring coefficient	c :	704	$[Nm/rad]$

For the SLIP and NSLIP simulations shown, except in Fig. 7, the system energy simulated is 1'860 Joules.

ACKNOWLEDGMENTS

We thank everyone who gave feedback during the writing of this manuscript. In particular, we appreciate the frequent and insightful discussions with Matthew Millard, Brent Gillespie and Andrea del Prete, as well as Friedrich Solowjow's advice on mathematical notation. We also appreciate the editors and reviewers for their constructive suggestions and quick turnaround time.

REFERENCES

[1] M. A. Daley and A. A. Biewener, "Running over rough terrain reveals limb control for intrinsic stability", *Proceedings of the National Academy of Sciences*, vol. 103, no. 42, 2006.

[2] M. A. Daley and J. R. Usherwood, "Two explanations for the compliant running paradox: Reduced work of bouncing viscera and increased stability in uneven terrain", *Biology letters*, vol. 6, no. 3, 2010.

[3] A. J. Ijspeert, "A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander", *Biological cybernetics*, vol. 84, no. 5, 2001.

[4] J. Proctor and P. Holmes, "Reflexes and preflexes: On the role of sensory feedback on rhythmic patterns in insect locomotion", *Biological cybernetics*, vol. 102, no. 6, 2010.

[5] D. Owaki, T. Kano, K. Nagasawa, A. Tero, and A. Ishiguro, "Simple robot suggests physical interlimb communication is essential for quadruped walking", *Journal of The Royal Society Interface*, vol. 10, no. 78, 2013.

[6] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, "Feedback control of dynamic bipedal robot locomotion", 2007.

[7] T. Koolen, S. Bertrand, G. Thomas, T. De Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a momentum-based control framework and application to the humanoid robot atlas", *International Journal of Humanoid Robotics*, vol. 13, no. 01, 2016.

[8] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation", in *IEEE 16th International Conference on Humanoid Robots*, 2016, pp. 339–346.

[9] Y. Zhao, B. R. Fernandez, and L. Sentis, "Robust optimal planning and control of non-periodic bipedal locomotion with a centroidal momentum model", *The International Journal of Robotics Research*, vol. 36, no. 11, 2017.

[10] E. Heijmink, A. Radulescu, B. Ponton, V. Barasuol, D. G. Caldwell, and C. Semini, "Learning optimal gait parameters and impedance profiles for legged locomotion", in *IEEE 17th International Conference on Humanoid Robots*, 2017.

[11] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. Atkeson, "Bayesian optimization using domain knowledge on the atrias biped", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1771–1778.

[12] R. Grandia, D. Pardo, and J. Buchli, "Contact invariant model learning for legged robot locomotion", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, 2018.

[13] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot", *The International Journal of Robotics Research*, vol. 32, no. 8, 2013.

[14] S. Rezazadeh, C. Hubicki, M. Jones, A. Peekema, J. Van Why, A. Abate, and J. Hurst, "Spring-mass walking with atrias in 3d: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot", in *Proceedings of the ASME Dynamic Systems and Control Conference*, 2015.

[15] S. W. Heim, M. Ajallooeian, P. Eckert, M. Vespignani, and A. J. Ijspeert, "On designing an active tail for legged robots: Simplifying control via decoupling of control objectives", *Industrial Robot: An International Journal*, vol. 43, no. 3, 2016.

[16] J. Ramos, B. Katz, M. Y. M. Chuah, and S. Kim, "Facilitating model-based control through software-hardware co-design", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 566–572.

[17] D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing, "Robotic vertical jumping agility via series-elastic power modulation", *Science Robotics*, vol. 1, no. 1, 2016.

[18] R. Tedrake, T. W. Zhang, and H. S. Seung, "Learning to walk in 20 minutes", in *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University New Haven, vol. 95585, 2005, pp. 1939–1412.

[19] S. Heim, F. Ruppert, A. A. Sarvestani, and A. Spröwitz, "Shaping in practice: Training wheels to learn fast hopping directly in hardware", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–6.

[20] R. Ringrose, "Self-stabilizing running", in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 1997, pp. 487–493.

[21] A. Schwab and M. Wisse, "Basin of attraction of the simplest walking model", in *Proceedings of the ASME design engineering technical conference*, vol. 6, 2001.

[22] H. Geyer, R. Blickhan, and A. Seyfarth, "Natural dynamics of spring-like running: Emergence of selfstability", in *5th International Conference on Climbing and Walking Robots (CLAWAR)*, 2002, pp. 87–92.

[23] J. Rummel and A. Seyfarth, "Stable running with segmented legs", *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 919–934, 2008.

[24] A. Wu and H. Geyer, "The 3-d spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments", *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1114–1124, 2013.

[25] T. McGeer, "Passive dynamic walking", *The International Journal Robotic Research*, vol. 9, pp. 62–82, 1990.

[26] M. Wisse and J. Van Frankenhuyzen, "Design and construction of mike; a 2-d autonomous biped based on passive dynamic walking", in *Adaptive motion of animals and machines*, Springer, 2006, pp. 143–154.

[27] P. A. Bhounsule, J. Cortell, and A. Ruina, "Design and control of ranger: An energy-efficient, dynamic walking robot", in *Adaptive Mobile Robotics*, World Scientific, 2012, pp. 441–448.

[28] F. Asano, "Stability principle underlying passive dynamic walking of rimless wheel", in *IEEE International Conference on Control Applications (CCA)*, 2012, pp. 1039–1044.

[29] A. D. Kuo, "Energetics of actively powered locomotion using the simplest walking model", *Journal of biomechanical engineering*, vol. 124, no. 1, 2002.

[30] R. Blickhan, "The spring-mass model for running and hopping", *Journal of biomechanics*, vol. 22, pp. 1217–1227, 1989.

[31] R. J. Full and D. E. Koditschek, "Templates and anchors: Neuromechanical hypotheses of legged locomotion on land", *Journal of Experimental Biology*, vol. 202, pp. 3325–3332, 1999.

[32] H.-M. Maus, S. Revzen, J. Guckenheimer, C. Ludwig, J. Reger, and A. Seyfarth, "Constructing predictive models of human running", *Journal of The Royal Society Interface*, vol. 12, p. 20 140 899, 2015.

[33] D. L. Jindrich and R. J. Full, "Dynamic stabilization of rapid hexapedal locomotion", *Journal of Experimental Biology*, vol. 205, pp. 2803–2823, 2002.

[34] R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, H. Brown, D. McMordie, U. Saranli, R. Full, and D. E. Koditschek, "Rhex: A biologically inspired hexapod runner", *Autonomous Robots*, vol. 11, no. 3, 2001.

[35] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer, "The dynamics of legged locomotion: Models, analyses, and challenges", *SIAM review*, vol. 48, no. 2, 2006.

[36] B. Stephens and C. Atkeson, "Modeling and control of periodic humanoid balance using the linear biped model", in *IEEE 9th International Conference on Humanoid Robots*, Dec. 2009, pp. 379–384.

[37] E. W. Hawkes and M. R. Cutkosky, "Design of materials and mechanisms for responsive robots", *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 359–384, 2018.

[38] R. M. Ghigliazza, R. Altendorfer, P. Holmes, and D. Koditschek, "A simply stabilized running model", *SIAM review*, vol. 47, no. 3, 2005.

[39] Ö. Arslan and U. Saranli, "Reactive planning and control of planar spring-mass running on rough terrain", *IEEE Transactions on Robotics*, vol. 28, no. 3, 2012.

[40] G. Piovan and K. Byl, "Two-element control for the active slip model", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 5656–5662.

[41] T. Cnops, Z. Gan, and C. D. Remy, "The basin of attraction for running robots: Fractals, multistep trajectories, and the choice of control", in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1586–1591.

[42] L. R. Palmer III and C. E. Eaton, "Periodic spring–mass running over uneven terrain through feedforward control of landing conditions", *Bioinspiration & biomimetics*, vol. 9, no. 3, 2014.

[43] M. Hutter, C. D. Remy, M. A. Höpflinger, and R. Siegwart, "Slip running with an articulated robotic leg", in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2010, pp. 4934–4939.

[44] L. Sentis, "Synthesis and control of whole-body behaviors in humanoid systems", PhD thesis, 2007.

[45] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model", in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2013, pp. 5134–5140.

[46] 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*, vol. 54, no. 8, pp. 1779–1793, 2009.

[47] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion", *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001.

[48] R. Altendorfer, D. E. Koditschek, and P. Holmes, "Stability analysis of a clock-driven rigid-body slip model for rhex", *The International Journal of Robotics Research*, vol. 23, no. 10-11, pp. 1001–1012, 2004.

[49] D. Renjewski, A. Spröwitz, A. Peekema, M. Jones, and J. Hurst, "Exciting engineered passive dynamics in a bipedal robot", *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1244–1251, 2015.

[50] W. C. Martin, A. Wu, and H. Geyer, "Experimental evaluation of deadbeat running on the atrias biped", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1085–1092, 2017.

[51] J.-P. Aubin, *Viability theory*. Springer Science & Business Media, 2009.

[52] J. N. Maidens, S. Kaynama, I. M. Mitchell, M. M. Oishi, and G. A. Dumont, "Lagrangian methods for approximating the viability kernel in high-dimensional systems", *Automatica*, vol. 49, no. 7, pp. 2017–2029, 2013.

[53] A. Liniger and J. Lygeros, "Real-time control for autonomous racing based on viability theory", *IEEE Transactions on Control Systems Technology*, no. 99, pp. 1–15, 2017.

[54] D. Panagou, K. Margellos, S. Summers, J. Lygeros, and K. J. Kyriakopoulos, "A viability approach for the stabilization of an underactuated underwater vehicle in the presence of current disturbances", in *Proceedings of the 48th IEEE Conference on Decision and Control (CDC)*, 2009, pp. 8612–8617.

[55] P.-B. Wieber, "Viability and predictive control for safe locomotion", in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 1103–1108.

[56] P. Zaytsev, W. Wolfslag, and A. Ruina, "The boundaries of walking stability: Viability and controllability of simple models", *IEEE Transactions on Robotics*, no. 2, pp. 336–352, 2018.

[57] D. Lakatos, W. Friedl, and A. Albu-Schäffer, "Eigenmodes of nonlinear dynamics: Definition, existence, and embodiment into legged robots with elastic elements", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1062–1069, 2017.

[58] N. Smit-Anseeuw, C. D. Remy, and R. Vasudevan, "Walking with confidence: Safety regulation for full order biped models", *arXiv preprint arXiv:1903.08327*, 2019.

[59] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances", in *IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 2242–2253.

[60] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models", *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[61] D. Fridovich-Keil, S. L. Herbert, J. F. Fisac, S. Deglurkar, and C. J. Tomlin, "Planning, fast and slow: A framework for adaptive real-time safe trajectory planning", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 387–394.

[62] A. Merker, D. Kaiser, A. Seyfarth, and M. Hermann, "Stable running with asymmetric legs: A bifurcation approach", *International Journal of Bifurcation and Chaos*, vol. 25, no. 11, p. 1 550 152, 2015.

[63] G. Piovan and K. Byl, "Reachability-based control for the active slip model", *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 270–287, 2015.

[64] S. Heim and A. Spröwitz, "Learning from outside the viability kernel: Why we should build robots that can fall with grace", in *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2018, pp. 55–61.

[65] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[66] D. Owaki and A. Ishiguro, "Enhancing stability of a passive dynamic running biped by exploiting a nonlinear spring", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 4923–4928.

[67] J. D. Karssen and M. Wisse, "Running with improved disturbance rejection by using non-linear leg springs", *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 531–539, 2011.

[68] C. D. Remy, K. Buffinton, and R. Siegwart, "A matlab framework for efficient gait creation", in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2011, pp. 190–196.

[69] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[70] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space", *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[71] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid", *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[72] T. Flayols, A. Del Prete, P. Wensing, A. Mifsud, M. Benallegue, and O. Stasse, "Experimental evaluation of simple estimators for humanoid robots", in *IEEE 17th International Conference on Humanoid Robots*, IEEE, 2017, pp. 889–895.

[73] R. Antonova, A. Rai, and C. G. Atkeson, "Sample efficient optimization for learning controllers for bipedal locomotion", in *IEEE 16th International Conference on Humanoid Robots*, 2016, pp. 22–28.

[74] V. C. Kumar, S. Ha, and K. Yamane, "Improving model-based balance controllers using reinforcement learning and adaptive sampling", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7541–7547.

[75] A. von Rohr, S. Trimpe, A. Marco, P. Fischer, and S. Palagi, "Gait learning for soft microrobots controlled by light fields", in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6199–6206.

[76] E. C. Whitman and C. G. Atkeson, "Control of a walking biped using a combination of simple policies", in *IEEE 9th International Conference on Humanoid Robots*, 2009, pp. 520–527.

Steve Heim received the B.Sc. degree in mechanical engineering and M.Sc. in robotics, systems and control from ETH Zurich, Switzerland, in 2012 and 2015, respectively. He then spent two years with the Ishiguro lab at Tohoku University in Sendai, Japan. He is currently pursuing the Ph.D. degree with the Dynamic Locomotion Group at the Max Planck Institute for Intelligent Systems in Stuttgart, Germany. His research interests include nonlinear dynamics, control and learning, particularly in relation to legged locomotion.

Alexander Spröwitz received the Diplom degree in mechatronics from Ilmenau Technical University in Germany in 2005, and the Ph.D. degree in manufacturing systems and robotics from the Biorobotics Laboratory at the Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland in 2010. Since 2016 he is the Max Planck Research Group Leader of the Dynamic Locomotion Group, and IMPRS-IS faculty at the Max Planck Institute for Intelligent Systems in Stuttgart, Germany. His current research focuses on the mechanisms underlying legged locomotion. Dr. Spröwitz and his team design and experiment with legged robots and models to infer biomechanics and neuro-control principles of motion in animals.

A Learnable Safety Measure

Steve Heim[1,*], Alexander von Rohr[2,3,*], Sebastian Trimpe[2] and Alexander Badri-Spröwitz[1]
[1]: Dynamic Locomotion Group, Max Planck Institute for Intelligent Systems, Germany
[2]: Intelligent Control Systems Group, Max Planck Institute for Intelligent Systems, Germany
[3]: Ingenieurgesellschaft Auto und Verkehr (IAV), Germany

Abstract: Failures are challenging for learning to control physical systems since they risk damage, time-consuming resets, and often provide little gradient information. Adding safety constraints to exploration typically requires a lot of prior knowledge and domain expertise. We present a safety measure which implicitly captures how the system dynamics relate to a set of failure states. Not only can this measure be used as a safety function, but also to directly compute the set of safe state-action pairs. Further, we show a model-free approach to learn this measure by active sampling using Gaussian processes. While safety can only be guaranteed after learning the safety measure, we show that failures can already be greatly reduced by using the estimated measure during learning.

Keywords: viability, safe learning, active sampling

1 Introduction

Learning control directly on hardware has great promise: learning would enable robots to adapt to changing environments, exploit un-modeled dynamics, as well as greatly decrease the engineering effort required to deploy a robot in the field. One of the challenges during the exploration process is that the system might visit failure states, such as a flying robot crashing or a legged robot falling over. These failure states can be costly in terms of time, damage to the robot or environment, and are often uninformative for the learning process. Unfortunately, the learning agent may not know a priori which actions lead to failure states. Furthermore, there may be actions which lead to unviable states: states which have not yet failed, but from which it is inevitable to reach a failure state in the future. Ideally, the learning agent only explores actions which keep the system within the set of viable states, also known as the *viability kernel* [1].

Although algorithms to compute conservative approximations of the viability kernel are available, they are contingent on accurate dynamics models, require substantial engineering effort and do not scale well for many types of systems [2, 3]. Alternatively, it can be useful to have a safety function which indicates how close the system is to leaving the viability kernel. Safety functions can help guide exploration to stay within the viability kernel without having to know its precise bounds. However, designing these functions is non-trivial, and faces the same issues commonly seen in designing reward functions: they are typically only approximate indicators of potential failures, require much handcrafting, and often introduce unwanted designer bias into the exploration.

We propose a model-free approach to learn a safety function, which captures the notion of viability without requiring the viability kernel to be explicitly computed. Our first contribution is a safety measure taken over the set of viable state-action pairs. Intuitively speaking, this measure describes the quantity of actions available that can avoid leaving the viability kernel. It therefore implicitly captures the structure of the systems dynamics, and how this relates to failure states, making it an effective safety function. Our second contribution is an algorithm for model-free learning of probabilistic estimates of both the measure and the viable set, using a Gaussian process (GP). On the one hand, making no model assumptions means we cannot guarantee safety until the measure has converged. We show, nonetheless, that the estimated measure can already be used during learning to reduce the number of visits to failure states significantly. On the other hand, keeping assumptions

[*]Equally contributing.

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

to a minimum allows this approach to be applied more readily to systems with arbitrary dynamics, where accurate models may be difficult to come by. This makes our approach particularly well-suited to systems that are difficult to model and where failures are costly but not critical.

1.1 Background and Related Work

Safe Learning with Viability Kernels and Back-reachable Sets. Two common model-based approaches to find safe sets are the computation of viability kernels [1, 3, 4, 5] and back-reachable sets [2, 6]. For viability, the user first defines a set of failure states; the viability kernel is then the set of states from which there exist actions, such that the system can avoid the failure set for all time. For back-reachability, a target set is defined; the back-reachable set is the set of states from which there exist actions, such that the system can reach the target set in finite time. In practice, these sets often coincide [5, 7, 8] and can be used interchangeably[2]. This depends, however, not only on the system dynamics but also on the specified failure and target sets. For example, the failure set for a walking robot may be defined as all states from which the robot cannot move its center of mass (e.g., it has fallen over and cannot recover), and the target set may be defined as reaching a specific location. If obstacles are blocking the path to the target location, the robot may be outside the back-reachable set of the target set, even though it is inside the viability kernel.

There are several algorithms used to compute back-reachable sets or viability kernels in state space; their effectiveness depends greatly on the assumptions used to model the system. For an overview, we recommend [2, 3]. To circumvent the difficulty of obtaining an accurate model from first-principles, models can also be learned from data. For example, Akametalu et al. [9] and Fisac et al. [10] learn a GP model of the dynamics of the system and disturbances, and use this to compute a conservative reachable set. As the system explores this set, the GP model is refined, and the set can (usually) be expanded. Fisac et al. [10] demonstrate their approach on quadcopter flight. They also point out the strong interdependence of safety and learning the systems true dynamics: safety guarantees are only as good as the models they are based on. In contrast, we do not model or learn the system dynamics, but a safety measure instead. We then estimate the set of viable state-actions directly from our measure, which enables model-free safe learning. Although this loses safety guarantees while learning the safety measure, it can be substantially easier to apply to complex systems.

Recently, the notion of viability has been extended to sets in state-action space. Zaytsev et al. [7] use this to directly link the reachable and viable sets. Heim and Spröwitz [11] use this to quantify the influence of system design on robustness to noisy actions, which is particularly relevant in learning control. We use the same notion of viability in state-action space, but extend the binary notion of viability (a state-action pair either belongs to the set or not) with a measure.

Bayesian Optimization and Reinforcement Learning with Safety Functions. Recently, safe Bayesian optimization (BO) using GPs has been used to apply model-free learning of controller parameters for systems with failure conditions [12, 13, 14]. In addition to modeling the controller performance, a second GP is used to model the safety of the controller parameter space. The safety model is used to restrict active sampling to parameters with a high probability of being safe. Though the controller parameters are applied to dynamical systems, safety is evaluated as purely dependent on parameter space, such that it can be considered as a static bandit problem. Thus, each sample of the parameter-space does not affect the safety of future samples. This approach is challenging to apply to situations that include non-steady-state behavior or where a set of controller parameters may be safe for some states but not others. In contrast to safe BO, we consider the more general case where safety is dependent on the current state. This emphasizes the role of the system dynamics, as they constrain the paths that can be taken through the search space. This type of problem can be modeled as a non-ergodic Markov decision process: that is, where not every state can be reached from every other state.

Turchetta et al. [15] have extended safe BO to Markov decision processes, and they demonstrate this on a non-ergodic grid-world example, where there exist states which are reachable, but from which the system cannot return. The notion of safety as ergodicity was previously formalized by Moldovan and Abbeel [16] in the general reinforcement learning context, who also point out the counter-intuitive result that more cautious exploration can often lead to faster convergence.

[2] This is the case when the viability kernel is ergodic.

In all of these approaches, it is assumed that a safety function can be sampled whenever visiting a new point in the search space (whether this is the parameter or state space). Safety is then inferred for nearby, unvisited points. The probable safety of these states can then be guaranteed using certain assumptions on the safety function, such as Lipschitz-continuity. However, this safety function is typically user-defined, and only indicative of what might cause failure. For example, Schillinger et al. [14] use the temperature of the engine at steady-state, Berkenkamp et al. [12] use a minimum performance threshold, and Turchetta et al. [15] use the ground inclination a rover needs to negotiate. Just as guarantees for model-based methods depend on the quality of the model, safe BO depends on a well-chosen safety function. In practice, the safety function is often chosen to be more conservative than strictly necessary. In contrast, our safety measure implicitly encodes the structure of the system dynamics and a definition of failure states. Furthermore, we show that the measure does not need to be known a priori, and can be learned in a model-free manner by sampling. With no model assumptions, safety guarantees can only be given once the measure has converged. Prior knowledge can, however, be introduced to reduce failures significantly.

The rest of the paper is structured as follows: In Section 2, we define all the necessary objects and introduce the safety measure. These concepts are illustrated with a toy example. In Section 3, we extend this to a probabilistic setting and present an algorithm to learn the safety measure in a model-free context. In Section 4, we show simulation results using our algorithm and point out key properties. In Section 5 we summarize our contribution and potential future work.

2 A Measure over the Viable Set

We consider systems with deterministic dynamics of the form $s' = T(s, a)$, where $s \in S$ is a state, $a \in A$ is an action, and T is the transition map of the dynamics to a new state s'. The set of failure states $S_F \subset S$ can be defined arbitrarily. For the sake of simplicity, we consider here a set of states from which there are no meaningful transitions and the system would need to be reset or replaced. We will define objects in the state-action space $Q := S \times A$. We use the shorthand $s' = T(q)$ where $q := (s, a) \in Q$. We will illustrate the defined objects on a discrete grid-world, amenable to pen-and-paper computation, and shown in Fig. 1.

Toy Model. Intuitively, the transition map in Fig. 1 can be thought of as representing a hovering spaceship affected by gravity, which is stronger near the ground. The spaceship can apply two levels of thrusters or allow itself to fall. The failure set is $S_F : \{5\}$, when the spaceship crashes.

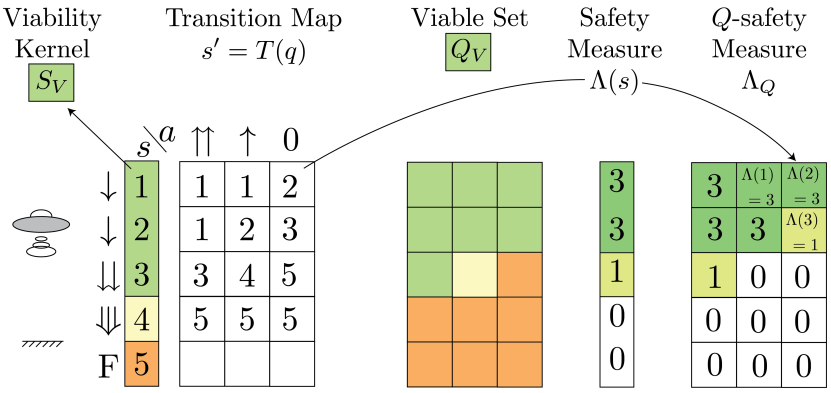


Figure 1: Shown are the transition map of our toy model, as well as each object: the viability kernel S_V, the viable set Q_V, the safety measure Λ and the Q-safety measure Λ_Q. Both S_V and Q_V are highlighted in green. We also highlight state-action pairs which result directly in failure in orange, and those that are unviable in yellow. The arrows and illustration are only to help with intuition.

We next define important mathematical objects for this work and illustrate them with the toy example. First, we define the viability kernel S_V.

Definition 1 (Viability Kernel)**.** *The* viability kernel $S_V \subset S \setminus S_F$ *is the maximal set of all states* $s \in S$*, from which there exists an action that keeps the system inside* S_V *(cf. [1, Chapter 1.1]).*

By its definition, states outside of S_V have either already failed, or will fail within finite time [1]. In the toy model, the viability kernel is $S_V = \{1, 2, 3\}$: for each of these states, there exists at least one action which can keep the spaceship from ever failing. Avoiding failure does not require ergodicity: the state $s = 3$ is viable, but it can no longer reach the other viable states. Note also that $s = 4$ is neither in the viability kernel S_V nor in the set of failure states S_F: it has not yet failed but cannot avoid reaching the failure set eventually. Next, we define the viable set in state-action space, Q_V.

Definition 2 (Viable Set)**.** *The* viable set $Q_V \subset Q$ *is the maximal set of all state-actions* q*, such that* $s' = T(q) \in S_V$.

By its definition, the viability kernel S_V is the projection of the viable set Q_V onto state space: for any state in S_V, the agent can sample a state-action from Q_V which maps back into itself [11]. Both S_V and Q_V are highlighted with green in Fig. 1. We can now define the safety measure Λ.

Definition 3 (Safety Measure)**.** *The* safety measure Λ *is the* n*-dimensional volume of the viable set* Q_V*. When applied to a point* $s \in S$*,* $\Lambda(s) \in \mathbb{R}_{\geq 0}$ *is the measure of the corresponding slice of* Q_V.

We use the Lebesgue measure for continuous spaces (assuming the sets are Lebesgue-measurable), and the counting measure for discrete spaces. Intuitively, a higher value $\Lambda(s)$ indicates that at state s, more viability-maintaining actions are available. A low value indicates that the agent should be very precise and deliberate since very few actions allow the system to avoid failure. In our toy model, for example, $\Lambda(3) = 1$ means that there is only one action which allows the system to avoid failure, this step and in the future. We can now map Λ into state-action space with the transition matrix.

Definition 4 (Q-Safety Measure)**.** *The* Q-Safety Measure Λ_Q *is defined as* $\Lambda(s')$ *where* $s' = T(q)$*. We use the shorthand* $\Lambda_Q(q) = \Lambda(s')$.

Next, we define safe level sets as the sets with measure $\Lambda_Q > \lambda$, where the minimum safety level λ is a non-negative scalar.

Definition 5 (Safe Level Sets)**.** *A safe level set* S_λ *is a set of states where* $\Lambda(s) > \lambda$*. A safe level set* Q_λ *is a set of state-action pairs which map into* S_λ*, such that* $\Lambda_Q(q) > \lambda$.

In other words, sampling from Q_λ will map the system to a state from which there is at least one action which maintains a safety level λ. Thus, the system can continue to choose actions which maintain a minimum safety level of λ indefinitely. In Fig. 1, the safe level-sets $Q_{\lambda=0}$ and $Q_{\lambda=2}$ are highlighted in different shades of green. This can be useful when certain types of disturbances are expected. We can recover the viable set from Λ_Q with $Q_V = Q_{\lambda=0}$, since viability implies $\Lambda(s) > 0$. Thus, if the safety measure Λ_Q is known, both the viable set Q_V and Λ can be obtained directly. If only Λ is known, Λ_Q can be computed directly using the transition map T. In the next section, we will use these facts to learn Λ_Q in a model-free fashion by sampling the dynamics.

3 Learning the Measure by Sampling

Given a system with an unknown transition map T and an unknown failure set S_F, our main objective is to estimate the viable set $\hat{Q}_V$ as a large, conservative approximation of the true viable set, $\hat{Q}_V \subseteq Q_V$. Since we assume an accurate dynamics model is unavailable, we directly sample the transition map T from a given state s by choosing an action a. Specifically, we begin sampling sequences from an initial state s. We then receive the tuple $(s', \textbf{\textit{failed}})$, where $s' = T(q)$ is the new state, and ***failed*** is a boolean indicating if $s' \in S_F$. The estimate $\hat{\Lambda}_Q$ can only be updated with the estimate Λ, except when sampling a failure. The goal is to choose actions a that are informative for learning the safety measure while avoiding the failure set S_F.

To achieve this goal, we model Λ_Q, from which we compute $\hat{Q}_V$ and $\hat{\Lambda}$. The estimate $\hat{\Lambda}_Q$ can already be used during learning to avoid actions with a low estimated probability of being safe.

3.1 Convergence Properties

To examine the requirements for $\hat{\Lambda}_Q$ to converge to the true measure Λ_Q, we separately consider the viable set Q_V and its complement Q_V^c, the set of unviable and failed state-action pairs.

Theorem 1. *Under the assumption of infinite random sampling over Q, the measure $\hat{\Lambda}_Q$ converges to the correct value 0 for all $q \in Q_V^c$.*

A proof for discrete state-action spaces can be found in appendix A. Once the measure $\hat{\Lambda}_Q$ inside Q_V^c has converged, the estimate for the viable set $\hat{Q}_V$ is tightly bounded from above. Therefore, the estimated safety measure is also tightly bounded to be $\hat{\Lambda} \leq \Lambda$. We can then ensure $\hat{\Lambda}_Q$ converges to the true measure by initializing optimistically, such that initially $\hat{\Lambda} \geq \Lambda$. These two conditions of infinite sampling and optimistic starts are typical for model-free learning [17, Chapter 2.6] but are also impractical. In particular, optimistic starts encourage visits to the failure set. We will now extend Λ_Q to a probabilistic setting, and use confidence bounds over the measure to estimate $\hat{Q}_V$. Although this loses the guarantee of converging to the true Λ_Q, we show that in practice it allows us to converge to conservative subsets while reducing failures effectively.

3.2 Probabilistic Estimates: Modeling Λ_Q with Gaussian processes

A probabilistic estimate allows us to (i) include prior knowledge without an explicit model of the dynamics, and (ii) estimate the uncertainty of the safety measure for a given state-action pair q, which we will use for active sampling. When modeling q as a random variable, the distribution should only allow for positive values and also have non-zero probability mass on the point zero, to model the probability of a point being unviable. We use a normal distribution as a practical approximation, where the probability mass below zero is treated as the discrete probability for the point zero. Specifically, we use a GP [18] to model the probabilistic estimate of Λ_Q. The posterior estimate of the measure at any point in Q is normally distributed, and it includes the prior assumptions on the estimate as well as the samples $\mathcal{D} = (q_i, \hat{\Lambda}_i(s_i'))$,

$$\hat{\Lambda}_Q(q)|\mathcal{D} \sim \mathcal{N}(\mu(q), \sigma^2(q))$$

where $\hat{\Lambda}_Q(q)|\mathcal{D}$ means the estimate is conditioned on the samples, $\mathcal{N}$ is the normal distribution, μ is the posterior mean function and σ^2 the posterior variance, given by the covariance function. The prior mean and covariance function can be used to encode the prior knowledge of the measure function, such as smoothness or known safe sets.

Given $\hat{\Lambda}_Q$, the probability that a state-action pair belongs to the safe level set Q_λ can be calculated using the cumulative distribution function of the normal distribution $F_{\hat{\Lambda}_Q}$ as

$$\mathbb{P}\left[\hat{\Lambda}_Q|\mathcal{D} > \lambda\right] \approx 1 - F_{\hat{\Lambda}_Q}\left[\lambda\right].$$

3.3 A Learning Algorithm for Λ_Q

We provide an approach for learning $\hat{\Lambda}_Q$ and the derived $\hat{Q}_V$ and $\hat{\Lambda}$, described in Algorithm 1. As discussed in Section 3.1, convergence requires an *optimistic* estimate of Q_V, such that the intitial estimate $\hat{\Lambda} \geq \Lambda$. Otherwise, a viable state-action pair may be incorrectly assigned the value 0. At the same time, the algorithm should use a *cautious* estimate for active sampling to reduce the probability of failing. To deal with this challenge, we use an optimistic set $\hat{Q}_{\text{opt}}$ to compute $\hat{\Lambda}$. A separate, cautious set $\hat{Q}_{\text{caut}}$ is used for active sampling. We obtain these as

$$\hat{Q}_{\text{opt}}(\gamma_{\text{opt}}) = \begin{cases} 1 & \text{if } \mathbb{P}\left[\hat{\Lambda}_Q|\mathcal{D} > 0\right] > \gamma_{\text{opt}} \\ 0 & \text{otherwise,} \end{cases}$$

$$\hat{Q}_{\text{caut}}(\gamma_{\text{caut}}, \lambda_{\text{caut}}) = \begin{cases} 1 & \text{if } \mathbb{P}\left[\hat{\Lambda}_Q|\mathcal{D} > \lambda_{\text{caut}}\right] > \gamma_{\text{caut}} \\ 0 & \text{otherwise} \end{cases}$$

by thresholding the probability with a minimum confidence $\gamma \in [0, 1]$. The algorithm has three tuning parameters: γ_{opt} governs the level of optimism in $\hat{Q}_{\text{opt}}$, and λ_{caut} and γ_{caut} govern the level of caution for active sampling. Choosing $\gamma_{\text{caut}} \geq \gamma_{\text{opt}}$ ensures that $\hat{Q}_{\text{caut}} \subseteq \hat{Q}_{\text{opt}}$, so we never purposefully explore outside the current estimate of the viability set. The algorithm samples the action from the cautious set $\hat{Q}_{\text{caut}}$ with highest variance. By actively reducing variance, the confidence in the measure is increased. Choosing actions with high variance also encourages exploration of the state space.

Algorithm 1 Learning the safety measure

Input: initial measure estimate $\hat{\Lambda}_Q$; thresholds γ_{caut}, γ_{opt}, and λ_{caut}; initial state s_0; maximum number of samples n.
while $i < n$ **do**
  Compute $\hat{\Lambda}$, $\hat{Q}_{\text{opt}}$ and $\hat{Q}_{\text{caut}}$ from $\hat{\Lambda}_Q$.
  $A_{caut} \leftarrow \forall a$ s.t. $(s_i, a) \in \hat{Q}_{\text{caut}}$. ▷ Determine safe actions.
  if A_{caut} is empty **then**
    $a_i \leftarrow \underset{(s_i,a)\in A}{\operatorname{argmax}} \mathbb{P}[a \in \hat{Q}_{\text{caut}}]$ ▷ Take safest action.
  else
    $a_i \leftarrow \underset{a\in A_{caut}}{\operatorname{argmax}} \sigma^2(s_i, a)$. ▷ Explore based on variance of the GP model.
  $(s_{i+1}, \textit{failed}) \leftarrow T(s_i, a_i)$. ▷ Sample the dynamics.
  if *failed* **then**
    Update $\mathcal{D}$ with $((s_i, a_i),\ 0)$ and recompute $\hat{\Lambda}_Q$
    $s_{i+1} \leftarrow$ random state from $\hat{Q}_{\text{caut}}$. ▷ Reset if failed.
  else
    Compute $\hat{\Lambda}(s_{i+1})$ from $\hat{Q}_{\text{opt}}$
    Update $\mathcal{D}$ with $((s_i, a_i), \hat{\Lambda}(s_{i+1}))$ and recompute $\hat{\Lambda}_Q$ ▷ Update measure estimate.

4 Results

We have tested our algorithm in simulation, and provide a Python implementation using the SciPy [19] and GPy packages [20]. The code can is available in the supplementary material and online at `github.com/sheim/vibly`, and includes example code to reproduce the results shown here, some additional examples, and a template to implement dynamics of other systems.

We report the results of two examples, which each highlight a specific challenge: dealing with unviable state-action pairs, and dealing with complex dynamics. We also use the second example to suggest guidelines for choosing the algorithm parameters, though this will typically be system-specific. Both examples are low-dimensional, and the ground-truth is computed by brute force. This allows us to easily choose reasonable parameters for the GP model, which is otherwise a separate challenge for using Gaussian processes. In practice, choosing these parameters is highly system-dependent [21]. We use a covariance function from the Matérn family [18, Chapter 4], which has two parameters: the length scales for each input dimension and the signal variance. The length scales describe how fast the measure changes when moving away from a known state-action pair. The second parameter is the signal variance, which relates to the total variation of the measure estimate $\hat{\Lambda}_Q$. Details for the models are in the appendix.

Unviable state-action pairs. Our first example is based on the hovership spaceship from Section 2. The model has been modified with a continuous state-action space, and the dynamics have been adjusted to increase the portion of the state-action space which is unviable. The GP prior mean is purposefully initialized poorly, such that most of the initial estimate $\hat{Q}_{\text{opt}}$ lies outside the true Q_V. This example shows that the algorithm can cope with unviable states, even though the ground truth is only sampled at failure. The confidence thresholds γ_{opt}, γ_{caut} and λ are initialized to encourage rapid initial exploration, then gradually increased to speed up convergence to a safe subset of Q_V. After 250 samples, it has nearly converged to the ground-truth, with an 8% failure rate (see Fig. 2c). In both examples shown in this paper, the confidence thresholds are increased linearly with each iteration as a heuristic that helps speed up convergence and reduces failures.

Complex, unmodeled dynamics. Our second example is a simulation of the spring-loaded inverted pendulum (SLIP) model, a low-dimensional idealized model commonly used to design controllers for running robots [6, 22, 23]. Control, and therefore learning, is applied once per step-cycle, at the apex of the flight phase. The system dynamics are therefore treated as a nonlinear, discrete map with a 2-dimensional state-action space; this nonlinear map is obtained by numerically simulating the full dynamics between two apex events. The set of failures, which includes falling and reversing direction, is evaluated on the full state space of the continuous dynamics. For this system, the measure Λ_Q has a non-smooth edge on the lower part of the state space, due to an infeasibility[3] constraint (see Fig. 3). Attempting to sample infeasible state-action pairs returns a failure. At this

[3] Infeasible state-action pairs have no physical meaning and cannot exist, such as starting underground.

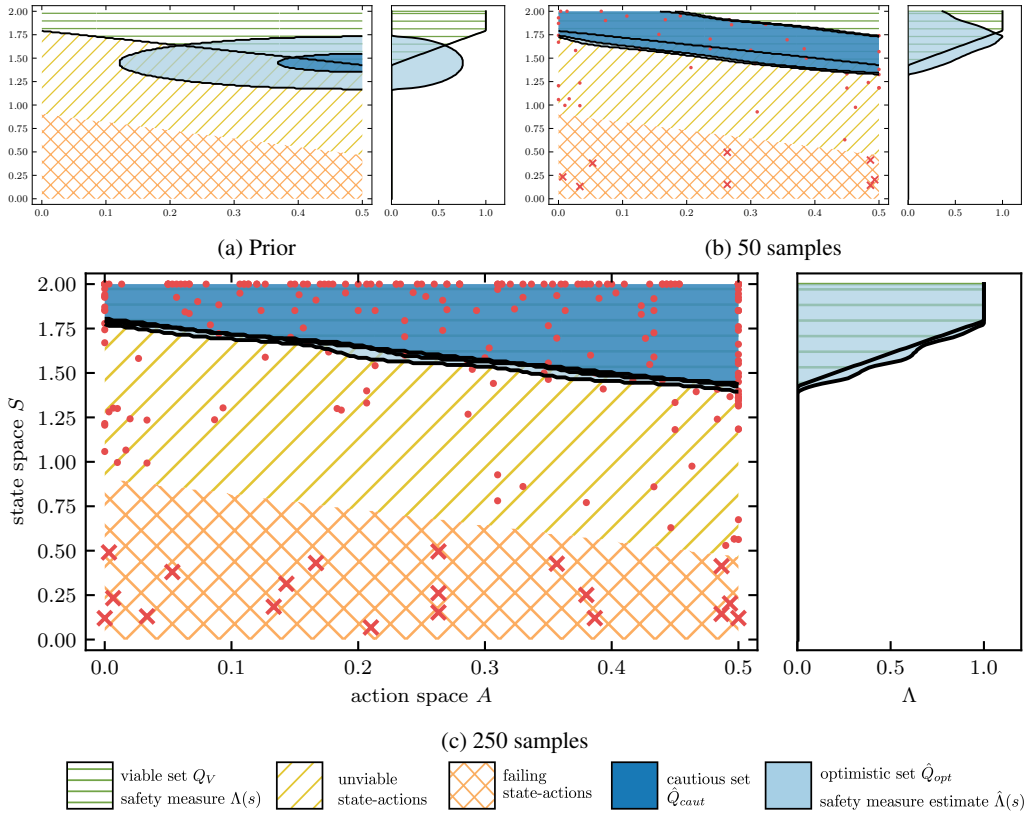


Figure 2: The progress of learning the $\hat{\Lambda}_Q$ for the spaceship model. Starting from an inaccurate prior 2a, after 50 samples 2b and after 250 samples 2c. Red dots indicate samples, red crosses samples that result in failures. After 250 samples, the optimistic set (light blue) and the measure Λ (green) are close to the ground truth. The cautious set $\hat{Q}_{\text{caut}}$ (dark blue) begins with a substantial region outside the viable set (2c), but quickly converges to the ground truth.

discontinuity, the smoothness assumptions encoded in the GP are violated. Therefore, more failures are sampled to learn the border of the viable set (see Fig. 3a). At the other borders of the viable set, where the smoothness assumption holds, the estimated sets approach the border despite sampling very few to no failures. When getting closer to the border of the viable set, the measure shrinks and the border of the set can be inferred without sampling unviable state-action pairs.

We also use this example to illustrate best practices for a realistic scenario, and the influence of different choices for the tuning parameters γ_{opt}, γ_{caut} and λ. The prior covariance function is obtained from simulations of an incorrect model, in which spring constant of the SLIP model is 20% lower. Since the covariance function encodes qualitative properties, it is reasonable to use a low fidelity simulation to obtain these GP parameters. The GP prior mean is typically more sensitive to simulation inaccuracies. It is therefore chosen around a known operating point, which we assume can be determined with conventional means without the need for a full model. Ideal operating points will feature a stable equilibrium-point or slow divergence from the operating point. Thus, the learning system can drive down variance locally before exploring more distant states, and the confidence bound γ_{opt} and γ_{caut} can be initialized more aggressively. We initialize the operating point of the SLIP model near a known limit-cycle of the running model. Although the prior is very conservative (see Fig. 3), the algorithm converges to a conservative yet nearly maximal approximation of the viable set Q_V, with a failure rate of 8% after 500 samples.

5 Discussion and Outlook

The first contribution of this paper is a safety measure taken over the set of viable state-action pairs. While this measure is useful in itself, computing viable sets relies on accurate models and is

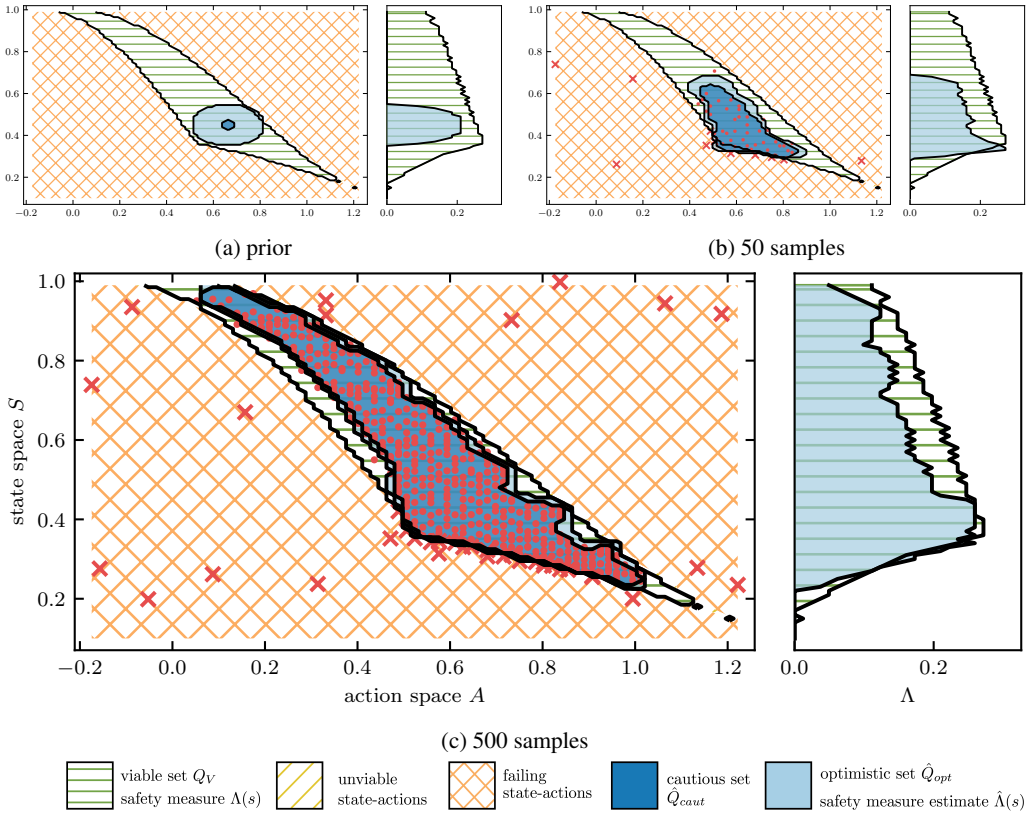


Figure 3: Learning the viable set and the corresponding measure $\hat{\Lambda}$ for a SLIP model. Starting from a conservative prior (3a), after 50 samples (3b) and after 500 samples (3c). The color coding is as in Fig. 2. A non-smooth infeasibility constraint bounds the bottom edge of the viable set. This violates the smoothness assumption of the GP, and requires many samples to learn accurately. The other edges are learned fairly accurately without many failures sampled. Actions close to the left edge are avoided, as they bring the system to states with low safety measure.

often intractable for systems with complex, high-dimensional dynamics. Our second contribution is a probabilistic, model-free approach to learn this measure and a safe set of state-action pairs using GPs. On the one hand, this makes it applicable to a variety of systems. On the other hand, making almost no assumptions means there are no hard guarantees for avoiding failure, even with a reasonable prior. This approach is therefore appropriate for systems which are difficult to model and where failures are costly but not critical, such as robots with soft or compliant components [24, 25] and small to mid-sized legged robots [21, 26, 27].

An issue with our current algorithm is that old samples contain old, potentially incorrect estimates of the measure, which can interfere with newer samples. A principled approach to only keep informative samples would improve the estimate and reduce computation costs. As with most other learning approaches, scaling to higher dimensions is a key challenge. An exploration strategy with an information-theoretic approach, especially with a heteroscedastic model (with state-dependent uncertainty), should improve both accuracy as well as sample-efficiency. In practice, it may be desirable to balance information gain of the safety measure and performance. How to balance this in a principled manner is an open question. We believe that leveraging a dynamics model will be key to scaling. How to map assumptions of the dynamics to the safety model requires further investigation. In addition, sample-efficiency might be greatly improved by updating all state-action pairs that are close in a dynamical sense instead of a Euclidean sense. How to obtain such a metric of closeness in state-action space is a problem we find is both challenging and has significant potential.

Acknowledgments

We thank Dominik Baumann, Matthias Neumann-Brosig and Friedrich Solowjow for insightful discussions during the project and while preparing the manuscript, as well as the anonymous reviewers for constructive and thorough feedback. We thank IMPRS-IS for the academic development of Steve Heim and Alexander von Rohr. This work was partly funded through the Cyber Valley Initiative and the Max Planck Society.

References

[1] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre. *Viability theory: new directions*. Springer Science & Business Media, 2011.

[2] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253, Dec 2017. doi:10.1109/CDC.2017.8263977.

[3] A. Liniger and J. Lygeros. Real-time control for autonomous racing based on viability theory. *IEEE Transactions on Control Systems Technology*, 27(2):464–478, March 2019. doi:10.1109/TCST.2017.2772903.

[4] P. Wieber. Viability and predictive control for safe locomotion. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1103–1108, Sep. 2008. doi:10.1109/IROS.2008.4651022.

[5] A. M. Bayen, E. Crück, and C. J. Tomlin. Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: Solving the hamilton-jacobi equation using viability techniques. In *Hybrid Systems: Computation and Control*, pages 90–104. Springer Berlin Heidelberg, 2002.

[6] G. Piovan and K. Byl. Reachability-based control for the active slip model. *The International Journal of Robotics Research*, 34(3):270–287, 2015. doi:10.1177/0278364914552112.

[7] P. Zaytsev, W. Wolfslag, and A. Ruina. The boundaries of walking stability: Viability and controllability of simple models. *IEEE Transactions on Robotics*, 34(2):336–352, April 2018. doi:10.1109/TRO.2017.2782818.

[8] S. Kaynama, J. Maidens, M. Oishi, I. M. Mitchell, and G. A. Dumont. Computing the viability kernel using maximal reachable sets. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '12, pages 55–64, New York, NY, USA, 2012. ACM. doi:10.1145/2185632.2185644.

[9] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin. Reachability-based safe learning with gaussian processes. In *53rd IEEE Conference on Decision and Control*, pages 1424–1431, Dec 2014. doi:10.1109/CDC.2014.7039601.

[10] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, July 2019. doi:10.1109/TAC.2018.2876389.

[11] S. Heim and A. Spröwitz. Beyond basins of attraction: Quantifying robustness of natural dynamics. *IEEE Transactions on Robotics*, 35(4):939–952, Aug 2019. doi:10.1109/TRO.2019.2910739.

[12] F. Berkenkamp, A. P. Schoellig, and A. Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496, May 2016. doi:10.1109/ICRA.2016.7487170.

[13] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint. Safe exploration for active learning with gaussian processes. In *Machine Learning and Knowledge Discovery in Databases*, pages 133–149. Springer International Publishing, 2015.

[14] M. Schillinger, B. Ortelt, B. Hartmann, J. Schreiter, M. Meister, D. Nguyen-Tuong, and O. Nelles. Safe active learning of a high pressure fuel supply system. In *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation*, pages 286–292, 2016.

[15] M. Turchetta, F. Berkenkamp, and A. Krause. Safe exploration in finite markov decision processes with gaussian processes. In *Advances in Neural Information Processing Systems 29*, pages 4312–4320. 2016.

[16] T. M. Moldovan and P. Abbeel. Safe exploration in markov decision processes. In J. Langford and J. Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1711–1718, July 2012.

[17] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction, 2nd Edition*, volume 1. MIT press Cambridge, 2019. URL http://incompleteideas.net/book/the-book-2nd.html.

[18] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[19] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed 23.09.2019].

[20] GPy. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy, since 2012.

[21] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. Atkeson. Bayesian optimization using domain knowledge on the atrias biped. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778, May 2018. doi:10.1109/ICRA.2018.8461237.

[22] P. M. Wensing and D. E. Orin. High-speed humanoid running through control with a 3d-slip model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5134–5140, Nov 2013. doi:10.1109/IROS.2013.6697099.

[23] C. Hubicki, J. Grimes, M. Jones, D. Renjewski, A. Spröwitz, A. Abate, and J. Hurst. Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot. *The International Journal of Robotics Research (IJRR)*, 35(12):1497–1521, 2016. doi:10.1177/0278364916648388.

[24] D. Surovik, K. Wang, M. Vespignani, J. Bruce, and K. E. Bekris. Adaptive tensegrity locomotion: Controlling a compliant icosahedron with symmetry-reduced reinforcement learning. *International Journal of Robotics Research (IJRR)*, 2019.

[25] D. Büchler, R. Calandra, B. Schölkopf, and J. Peters. Control of musculoskeletal systems using learned dynamics models. *IEEE Robotics and Automation Letters*, 3(4):3161–3168, Oct 2018. doi:10.1109/LRA.2018.2849601.

[26] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246, Oct 2018. doi:10.1109/IROS.2018.8593722.

[27] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi:10.1126/scirobotics.aau5872.

A Convergence of the measure estimate for unviable state-action pairs

We will show here the convergence result for systems with discrete time and finite, discrete state-action spaces. Specifically we show that, under the assumption of inifinite sampling, the estimated measure $\hat{\Lambda}_Q$ for the unviable state-action pairs $q \in Q_V^c$ converges to the true value of 0 when following the updates

$$\hat{\Lambda}_Q(q) = \begin{cases} 0 & \text{if } \textbf{\textit{failed}} \\ \hat{\Lambda}(s') & \text{else.} \end{cases} \tag{1}$$

A direct consequence is that the estimated measure $\hat{\Lambda}$ for all unviable states also converges to the true value of 0. This provides an upper bound for the measure. We proceed to the theorem.

Theorem 1. *Under the assumption of infinite random sampling over Q, the measure $\hat{\Lambda}_Q$ converges to the correct value 0 for all $q \in Q_V^c$.*

We define $S_U = (S_V \cup S_F)^c$ as the set of unviable states, and $Q_U \coloneqq S_U \times A$. We also define the operator $\mathrm{len}(s)$, which returns the integer length of the longest trajectory starting from the state s. We begin by showing that this theorem holds for all $q \in Q_U$, which ensures that the estimated measure converges to 0 for all $s \in S_U$. Consider all possible trajectories starting from any state $s \in S_U$. By the definition of viability, they are all *acyclic*, meaning no state is ever visited more than once. They therefore all end in S_F within finite time.

Lemma 1. *The longest trajectory starting from any state $s \in S_U$ has length $\mathrm{len}(s) \leq n$, where n is the number of states in S_U.*

This can be proven by contradiction. Let us assume that the longest trajectory starting in S_U has length $n_{longest} > n$. We take a sub-trajectory of length n; due to the acyclicity condition, this trajectory has has visited n unique states, and therefore has visited all states in S_U. It therefore cannot be lengthened without breaking the acyclicity condition, contradicting our assumption.

Lemma 2. *For every $i = 1, \ldots, n_{longest}$, there exists at least one state $s \in S_U$, for which the longest trajectory beginning from that state has length i.*

Again, this can be shown by contradiction. Let us assume there are no states with $\mathrm{len}(s) = 1$. Then take the longest trajectory starting from any $s \in S_U$, and proceed to the last state in the trajectory. By our assumption, $\mathrm{len}(s) > 1$, which implies that there is at least one action available from this state which would avoid failure and therefore increase the length of the trajectory by at least 1, contradicting the our previous statement. This reasoning can be extended to all other i up to $n_{longest}$ by inserting shorter states in the failure set S_F and repeating this process.

Proof. Now it is clear that sampling any q from a state with $\mathrm{len}(s) = 1$ will immediately transition to S_F, and therefore will be updated with the ground truth as per 1. Once each of such q has been sampled once, the measure estimate will be 0 for any state with $\mathrm{len}(s) = 1$. At this point, sampling any q from a state with $\mathrm{len}(s) = 2$ will be updated with 0, and so on until all $s \in S_U$ have converged to 0.

We can now turn our attention to the remaining $q \in Q_V^c$. By definition, these are state-action pairs that transition to S_U in a single step. Therefore, as soon as the estimated measure for all $s \in S_U$ has converged to 0, these will also be updated correctly with $\hat{\Lambda}(s) = 0$. □

B Dynamics of Simulated Examples

We include here the details of the dynamics for the systems used in Section 4. The implementation in Python is available in the supplementary material. We also include in the supplementary material code to compute the true viable sets, although this is only computationally tractable for small systems.

B.1 Hovering Spaceship

This example is a hovering spaceship loosely based on the toy example in Section 2, with a continuous state-action space. The spaceship has a single state, a vertical position, and is affected by nonlinear gravity. The non-constant gravity has been adjusted to accentuate the issue of nonviable which have not yet failed. The dynamics are:

$$\dot{s} = g_0 + \tanh(0.75\, s)g - a \tag{2}$$

where s is the height, g_0 is a baseline gravitational acceleration, and g is a coefficient for the gravity which increases with state. The spaceship can counteract gravity with the action $0 \leq a \leq a_{max}$. The failure set is defined as $s \geq s_{max}$. We also model a control frequency ω, such that the spaceship can only choose a new thrust a once every $\frac{1}{\omega}$ seconds. This control delay further accentuates the unviable states. The reader is encouraged to adjust these parameters in the code to see how this affects both the viable sets, and the learning of the safety measure.
The parameters used to generate the graph in the paper are:

base gravitational acceleration	g_0 :	0.1
gravitational acceleration	g :	1
max thrust	a_{max} :	0.5
ground height	s_{max}:	2
control frequency	ω :	1

B.2 Spring Loaded Inverted Pendulum

The spring-loaded inverted pendulum is a common model for understanding running dynamics, both in biomechanics and robotics. The body is represented by a point-mass, and a massless spring represents the leg. It has hybrid dynamics, meaning the governing equations of motion switch between different phases, and cyclic orbits. We begin each cycle at the apex, the highest point during flight phase, when vertical velocity is zero. Each step cycle has 3 phases: A flight phase terminating with a touchdown event, a stance phase terminating with a liftoff event, and a second flight phase terminating with an apex event.
Between two apex events, we integrate the full state $[x, y, \dot{x}, \dot{y}]^\intercal$, where x and y are the horizontal and vertical positions of the point-mass. During each flight phase, the dynamics are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \end{bmatrix},$$

where g is the gravitational acceleration. During stance phase, the dynamics are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \frac{k\,(l_0 - l)}{m} \begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix} - \begin{bmatrix} 0 \\ g \end{bmatrix}$$
$$\theta = \arctan 2\left(\frac{y}{x}\right) - \frac{\pi}{2}$$
$$l = \sqrt{(x^2 + y^2)},$$

where k is the spring stiffness, l_0 is the spring resting length, and m is the mass. For concise notation, the reference frame is centered on the foot during stance. In the implementation, the foot position is also tracked and accounted for. The events are detected with

$$\text{touchdown: } l = l_0$$
$$\text{liftoff: } \theta = \arctan 2\left(\frac{y}{x}\right) - \frac{\pi}{2}$$
$$\text{apex: } \dot{y} = 0.$$

For simplicity, we can examine the state once per step cycle, on the so-called Poincaré section, at the apex of flight. At apex, all the potential energy at apex is contained in the height of the point-mass, and the kinetic energy in the forward velocity (the vertical velocity is zero by the definition of apex). Since the system is energy-conservative, we can use the potential energy normalized by total energy as our single state. We thus use as state $s = \frac{E_{\text{pot}}}{E_{\text{pot}}+E_{\text{kin}}} = \frac{gy}{\frac{\dot{x}^2}{2}+gy}$, where E_{pot} and E_{kin} are the potential and kinetic energy, respectively. For our simulations, we also define a single action: the landing angle of attack of the leg, $a = \alpha$, which is instantly reset to the desired angle at each apex.

For the figures in the paper, we used the following parameters, which are similar to human averages:

gravitational acceleration	g :	9.81	$[m/s^2]$
body mass	m :	80	$[kg]$
spring stiffness	k :	8200	$[N/m]$
spring resting length	l_0:	1	$[m]$

Learning from Outside the Viability Kernel: Why we Should Build Robots that can Fall With Grace

Steve Heim and Alexander Spröwitz

Abstract— **Despite impressive results using reinforcement learning to solve complex problems from scratch, in robotics this has still been largely limited to model-based learning with very informative reward functions. One of the major challenges is that the reward landscape often has large patches with no gradient, making it difficult to sample gradients effectively. We show here that the robot state-initialization can have a more important effect on the reward landscape than is generally expected. In particular, we show the counter-intuitive benefit of including initializations that are *unviable*, in other words initializing in states that are doomed to fail.**

I. INTRODUCTION

Recent advances in reinforcement learning (RL) have shown a lot of promise, especially with the capability to learn complex policies from scratch, model-free and from very generic reward signals [1][2]. However it has been difficult to transfer these results into learning directly in robot hardware. Most RL in robotics is model-based and uses highly informative reward functions [3].

One of the main challenges is that much of the reward landscape (sometimes called a cost landscape) that a learning algorithm needs to traverse tends to be flat, providing no informative gradient. In classical RL, the ability to run a huge number of trials, often in parallel, with completely random initial conditions for both state and policy parameters helps alleviate the problem. With a robot, this can be prohibitively expensive in terms of time, hardware costs and computation. It is typically necessary to invest substantial effort into the design of a reasonable initial controller and parameterization, and subsequently improve it via RL [4][5], instead of learning policies from scratch.

Another common approach is to rely on model-based control methods, and use RL to learn the model, instead of learning the policy [3][6].

It is also important to choose an appropriate exploration strategy, and the subfield of intrinsic motivation in particular addresses the problem of getting stuck in a flat portion of the reward landscape [7]. Alternatively, the reward landscape can be shaped by providing more informative reward functions [8][9], or a progressive set of reward functions or environments [10][11][12][13].

While these approaches all have merit, we show that the choice of state initialization can have a greater effect on the reward landscape than is usually assumed. The current practice is to initialize from a stable state, as states that fail with a high probability can be crippling. Quick failures often result in no reward and therefore no gradient to learn from, and often damage the robot hardware itself. At the most restrictive, this means the robot always starts within the basin of attraction of a stable controller. In order to accelerate learning, it is important for the learning agent to explore outside this region, and indeed considerable effort has been made to be able to step outside the basin of attraction and into its superset [14][15], the *viability kernel*.

A viability kernel [16] is the set of all states from which there is at least one time-evolution that remains confined to a desired region. Sampling states outside the current policy's basin of attraction allows the agent to learn more aggressively and still avoid potentially disastrous failures. When learning a model of the dynamics, the agent can clearly benefit by being even more aggressive and initially visit failure states. These states will often be in areas of the state-space that are otherwise not visited, and data-points sampled there can give additional information to fit a model more accurately. When directly learning a policy in a model-free approach however, the benefit of visiting states from which no stabilizing policy exists is less obvious.

The main contribution of this work is to show that it can be beneficial for model-free learning to initialize the robot from states *outside the viability kernel*.

II. VIABILITY KERNEL OF A RUNNING MODEL

A. Model

Our work revolves around the spring-loaded inverted pendulum (SLIP) model, a hybrid dynamic model ubiquitous in both biomechanics and the robotic legged-locomotion community for modeling running or hopping gaits. This model can be fit very accurately to experimental data of many different running animals [17][18], allows accurate prediction [19], and also has been used to design controllers for simulations [20][21][22][23] as well as actual robots [24][25][26]. Indeed, there has been a lot of effort to give legged robots SLIP-like behavior, either through mechanical design [27][28] or control [24][29]. The two dimensional view of the model, in the sagittal plane parallel to the direction of travel, represents a submanifold of the 3D-space, and the results can be extended to 3D motion both in simulation [23] and hardware [30].

We consider this an ideal model that is both low-dimensional enough to clearly illustrate our result, while also informative and applicable to real-world systems. The model, shown in Fig. 1, has a point mass representing the body center of gravity, and a massless spring representing the leg. Touchdown and liftoff conditions govern the switch

Dynamic Locomotion Group, Max Planck Institute for Intelligent Systems, Stuttgart, DE {heim, sprowitz}@.is.mpg.de

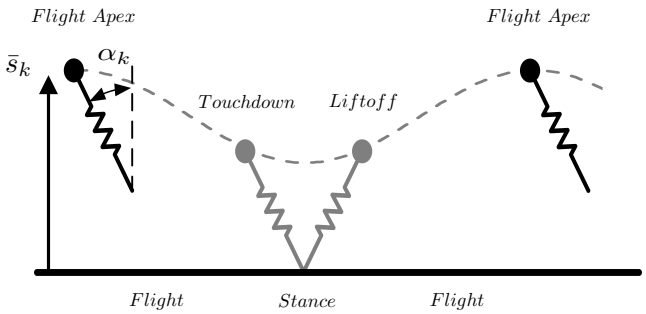


Fig. 1: The classical spring-loaded inverted pendulum is characterized by hybrid dynamics: the governing equations of motion switch between flight and stance phases at the touchdown and liftoff events. We have highlighted here the normalized height at apex $\bar{s}_k$ as our one-dimensional state and the landing angle of attack α_k as our one-dimensional action.

between flight and stance phases. The model uses a no-slip assumption for the foot during stance phase. At liftoff, the massless leg is instantaneously reset to the landing angle of attack α_k. The continuous dynamics are as follow:

Continuous Dynamics

Flight Dynamics

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \end{bmatrix}$$

Stance Dynamics

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \frac{k}{m}(l_0 - l)\begin{bmatrix} \sin(\alpha) \\ \cos(\alpha) \end{bmatrix} - \begin{bmatrix} 0 \\ g \end{bmatrix}$$

Touchdown Condition

$$y = l_0 \cos(\alpha_k)$$

Liftoff Condition

$$l = l_0$$

$$\alpha = \arctan(y/x) - \frac{\pi}{2}$$
$$l = \sqrt{((x - x_f)^2 + y^2)}$$

x : horizontal position	l_0 : leg resting length = 1 $[m]$
y : vertical position	k : spring stiffness = 8200 $[N/m]$
α : angle of attack	α_k : landing angle of attack $[rad]$
x_f : foot position	m : body mass = 80 $[kg]$
l : leg length	g : gravity constant = 9.81 $[m/s^2]$

The parameters used are parameters commonly used to model human running [31].

As is commonly done [20][21][32], we use Poincaré return maps to study the system [33]. We numerically integrate the continuous-time dynamics from one apex height to the next, and examine the state at these apex events. This gives us the following discrete mapping:

Discrete Dynamics

Continuous Dynamics Integrated from Apex to Apex

State at Apex

$$s_k = \begin{bmatrix} y_{apex} \\ \dot{x}_{apex} \end{bmatrix}$$

Normalized Height at Apex

$$\bar{s}_k = \frac{E_g}{E_g + E_k} = \frac{g\ y_{apex}}{\frac{\dot{x}_{apex}^2}{2} + g y_{apex}}$$

Apex Transition Dynamics

$$\bar{s}_{k+1} = P(\bar{s}_k, \alpha_k)$$

E_g : Potential Energy

E_k : Kinetic Energy

P : Poincaré map

We dropped the x coordinate as it is not periodic, and at apex the $\dot{y}$ coordinate is zero by definition, which leaves us with only two coordinates, y and $\dot{x}$. Finally, we exploit the fact that the system is energy-conservative and use the constant total energy constraint to reduce the system to a single state, a normalized hopping height. This results in the map $\bar{s}_{k+1} = P(s_k, \alpha_k)$. Note that we include the landing angle of attack parameter α_k as a control input which can be chosen freely at each apex. The original passive model was used to analyze steady-state locomotion and set α_k as a constant parameter; for non-steady-state locomotion, the landing angle of attack is a well studied choice for a control input [23][30][34], alongside spring-stiffness.

B. Transition Matrix and Viability Kernel

The discrete dynamics of the system have two possible transitions: either a mapping from one apex height to the next apex height, or from an apex height to a failure state, in which the point-mass body hits the ground with $\bar{s}_{k+1} = 0$.

Having reduced both state and action space to a single dimension, we obtain the entire transition matrix by brute force simulation of a finely discretized grid over $\bar{s}$ and α_k, use this to compute a difference in state and visualize the result in Fig. 2. The grey regions are failures ending with the body hitting the ground, and the colored regions are all the state-action combinations that result in a second hop. The warm colored region represent actions that lead to a higher height at the next apex, whereas the cold-colored region are actions that result in a lower height at the next apex. The intersection between these two regions are limit-cycles where the apex height remains constant. As the end-goal of locomotion is arguably to reach a specific distance, actually traveling on a limit-cycle is not strictly necessary: a policy which hops

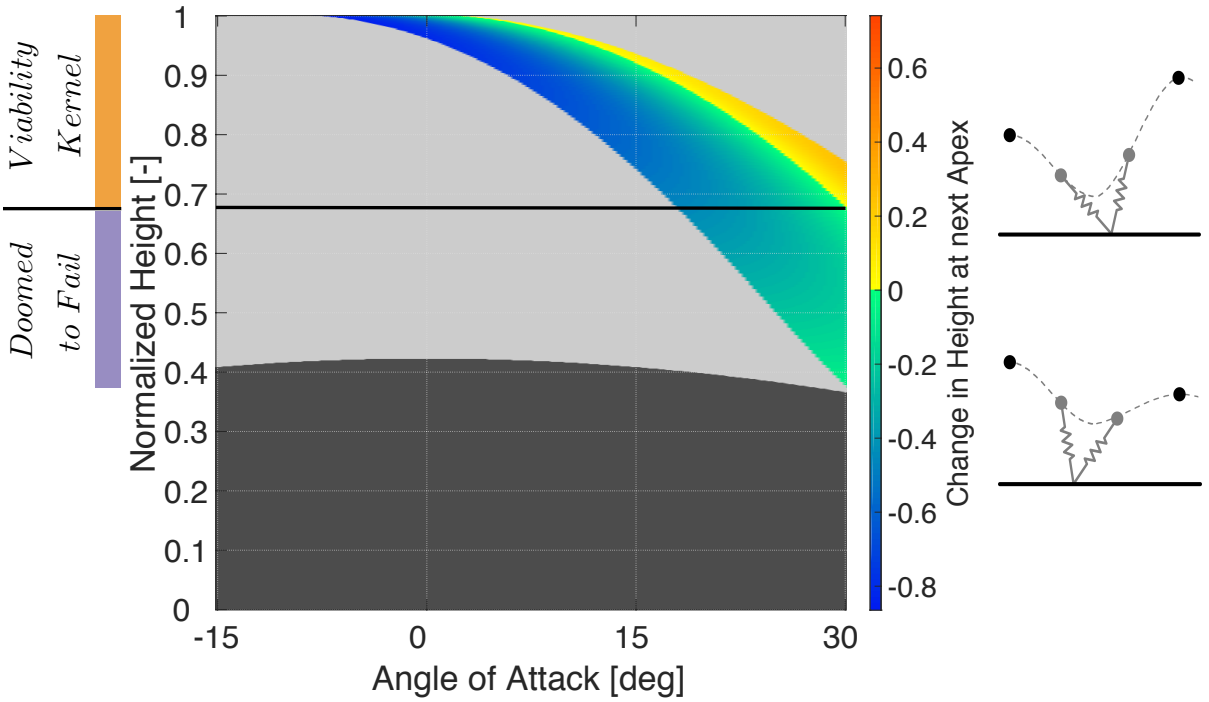


Fig. 2: Represented is the change in state for the SLIP model, with an energy content of $E_g + E_k = 1.86kJ$. The grey regions represent state-action pairs that result in falls, and the black regions are infeasible points: the foot would start underground at apex. The warm colored region result in a higher apex height, whereas the cold colored region results in a lower apex height. The black line at normalized height 0.675 indicates the lower bound of the viability kernel: any states below this height can only select actions that lead to a lower hopping height, until it inevitably falls.

erratically can also reach the end-goal. The black region represents state-action combinations that are infeasible: the foot would start underground at apex.

This constraint between 0° and 30° for α_k is convenient to illustrate our case, while also corresponding to the usual range of angles used in human running [31]. It is also a realistic constraint a robot might need to cope with, for example due to mechanical hard-stops, limitations in hip-swing velocity or to remain within friction cones.

The viability kernel of this constrained SLIP-model can be seen by inspecting Fig. 2, and is the set of normalized apex heights $\bar{s}_k \in [0.675,\ 1]$. For each state in this set, actions can be chosen that either keep the system on a limit-cycle, bring it to a higher, or to a lower apex height. Thus, it is always possible to stay inside the set, making it a viability kernel [16]. Any state that starts below the normalized height threshold of 0.675, marked with a horizontal line in Fig. 2, can only choose actions that either immediately fall or at best hop at subsequently lower apex heights until it falls. These states are *doomed to fail*.

III. Control and Learning

In our study, our goal is to learn a control policy for choosing α_k at each apex which travels as far as possible from any viable initial state without falling.

A. Control Policy

Various control policies have been proposed for SLIP models, often emphasizing deadbeat control [23][34] or some form of optimality [20][35]. At its simplest, even a linear controller is able to stabilize any of the limit cycles, even if not in an optimal manner. We limit this study to a linear controller, visualized in Fig. 3, since its low dimensionality allows us to directly visualize the reward landscape in parameter space, as in Fig. 4. The core of our results depends on the structure of the transition matrix (the dynamics) both inside and outside the viability kernel, which is independent of the parameterization used.

Specifically, we use a linear gaussian policy

$$\alpha_k \sim \mathcal{N}(\mu, \sigma^2)$$
$$\mu = \theta_0 s_k + \theta_1$$

where $\mathcal{N}$ is the normal distribution, μ is the mean and represents the greedy policy in absence of exploration, and σ^2 is the variance (our exploration parameter). The parameters θ_0 and θ_1 are the slope and offset of our linear policy[1]. Gaussian policies effectively include exploration directly into the policy and are often effective in continuous state and action systems, as often encountered in robotics [2][3].

[1]Though technically affine, we use the common practice of calling this linear to avoid confusion.

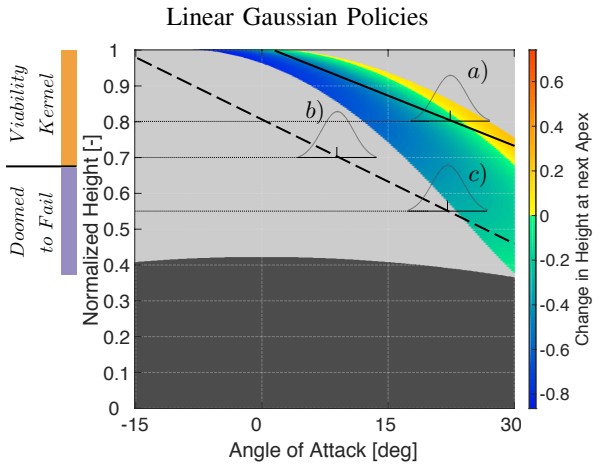


Fig. 3: Represented are two linear controllers. The solid line is a well-tuned controller that would effectively stabilize the system, whereas dashed line is a guess that has the correct general shape, but is far from the salient gradient set (SGS). To highlight the problem, we have overlaid the gaussian sampling for three possible situations: in *a*) the gaussian policy would sample with high probability from the SGS for a state inside the viability kernel. With the second greedy controller in *b*) however, the probability of sampling a non-failing action is close to 0. Case *c*) shows that the second greedy controller has a decent chance of sampling successfully from the SGS when initialized outside the viability kernel. It can learn from these samplings even though all initializations are doomed to fail.

B. Interplay of Parameter Initialization, State Initialization and Exploration Strategy

As an illustrative example, we award a fixed reward at each apex, which equates to learning to hop as many times as possible, regardless of the initial conditions. Using other periodic rewards, such as distance traveled per step, does not change the quality of the results.

It is clear that in order to learn, the agent needs to sample at least once a state-action pair that transitions to another apex: if the agent immediately samples from the state-action pairs that result directly in a fall, colored grey in Fig. 2, it will have a constant reward of 0 and no gradient to learn from. For convenience, we will call the set of state-action pairs that result in a second apex height a *salient gradient set (SGS)*.

With a random initialization of the policy parameters θ_0 and θ_1, it is possible that the greedy policy lies completely outside the SGS of the transition matrix, and therefore has a low chance of sampling from the SGS. The standard solution of increasing the variance of the gaussian policy has a drawback. While it increases the probability of sampling from the SGS when the greedy policy is initialized far away from the SGS, it also increases the probability to sample failing actions when the greedy policy is well tuned, as visualized with the solid line in Fig. 3.

In this work, we highlight another aspect of the problem which, to the best of our knowledge, has been largely overlooked. In fig. 3 we show an example where the greedy policy chooses an action far from the salient gradient set for any state belonging to the viability kernel. In these cases, even local exploration will have only a low probability of ever taking a second step and receiving a reward. The same policy has a high probability of sampling non-failing actions for states *outside* the viability kernel. In other words, we can directly change the reward landscape by including these states in the state initialization.

C. Landscapes

We discretized over the parameters θ_0 and θ_1, computed 100 roll-outs for each point of the grid using a fixed exploration rate σ and used the average return as an estimate of the reward. We then interpolated over the grid to obtain the reward landscapes.

We repeated this for two scenarios: strictly *viable initializaiton* (left column in Fig. 4) and *feasible initializations* (right column in Fig. 4. Viable initializations are initial states sampled with a uniform distribution from the viability kernel, whereas feasible initializations are sampled from a uniform distribution of all feasible states, including states that are doomed to fail.

We have also repeated this estimate for different variance, to highlight the much greater effect the state initialization has on the reward landscape. Indeed we found that using feasible initialization increased the SGS by just over 79%. Nearly doubling the variance, from 8° to 15°, only increased the SGS by 13%. This highlights a case where a large effort to find effective exploration strategies can be replaced with an improved state initialization.

D. Learning Setup and Results

To test our results, we implemented a standard temporal-difference algorithm with eligibility trace, a typical policy-gradient class of reinforcement learning algorithms [2][36] relying on the likelihood ratio [37] to estimate the policy gradient. To find appropriate hyperparameters for the learning step size and discount factor, several hundred learning trials were run with randomized hyperparameters, which revealed most consistent performance with a very low discount factor around 0.95, and learning step sizes on the order of magnitude of 0.001; in the presented results, these values are used. The variance of the gaussian policy begins at 15° and is reduced to 10°, 7.5° and 5° as the average performance reaches 2, 3 and 4 steps respectively. Once a policy averages at least 15 steps, learning is terminated.

To compare learning performance we ran learning trials starting from the same initial policy parameter, using first only viable initializations and then feasible initializations. In the second case, once the policy averages more than three steps, we consider the policy to have learned sufficiently and switch to a viable initialization, since the end-goal is to be stable for viable initializations.

We first ran trials for 50 randomized policy initializations (see Fig. 5). Before accepting each randomly selected policy initialization, its performance is estimated by averaging the reward over 100 roll-outs with viable initialization, as we did when estimating the reward landscape. Policy parameters

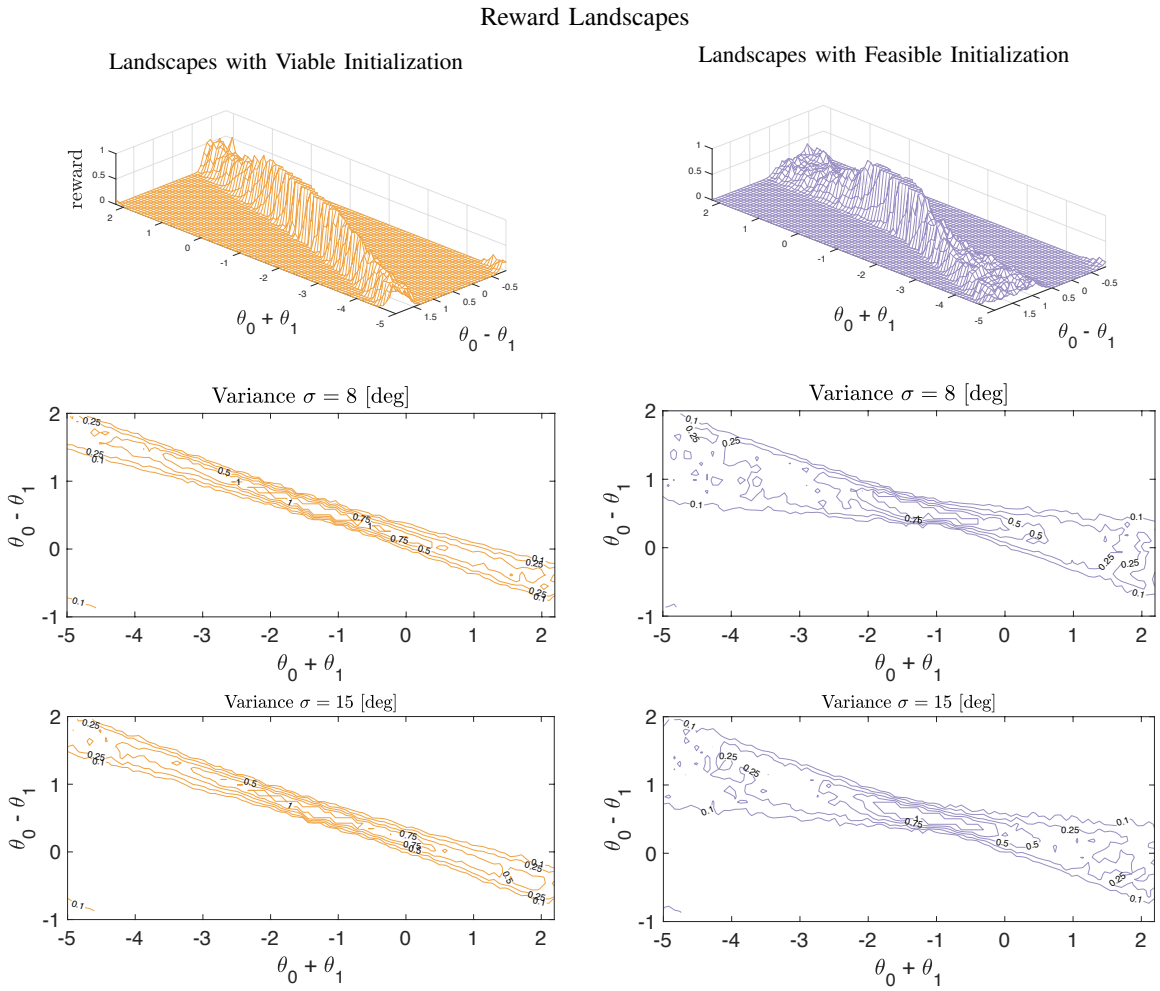


Fig. 4: In the reward landscapes on the left, the system is initialized with a uniform random sampling inside the viability kernel, whereas on the right it is initialized from states with a feasible action, whether viable or doomed to fail. For visual clarity, the parameter space has been rotated 45 degrees and the reward (number of steps taken) is capped at 1. At values greater than this, the gradient is very steep and policies quickly become stable for many steps. The meshed landscapes (top row) are shown for clarity and are the same as the first set of contours (second row) with variance of 8°. The landscape on the right has a salient gradient set (a point in parameter space with non-zero gradient) just over 79% larger. The landscapes with a more aggressive variance of 15° (bottom row) are shown to highlight how little effect increasing exploration has in comparison to the choice in state initialization.

with less than 0.5 steps on average are discarded and re-sampled. In other words, we biased initializations towards policy parameters for which viable initializations also had a chance to learn, instead of including policy parameters where viable initializations had no chance of learning.

As shown in Fig. 5, we found that starting with feasible initializations resulted in 60% more trials ending in success. The policy initializations relative to the landscapes is shown in Fig. 6. We then selected a single parameter at random, and repeated 10 trials each with viable and feasible initializations, shown in Fig. 7. Again, we found much greater reliability in learning success when starting from feasible initialization rather than strictly viable initializations.

IV. Discussion and Outlook

We show the effect state initialization has on the reward landscape of a dynamically unstable system, and a clear case where a naive policy initialization learns more reliably when unviable states, that is states that are doomed to fail, are included in the initializations. We have in particular highlighted it is to consider the effect of state initialization in overcoming a common problem in robot learning: reward landscapes with large patches devoid of gradient information. This adds another tool to design learning setups, in addition to more established approaches of exploration strategies or reward shaping.

One of the major drawbacks of initializing in unviable states is that failures often result in damage to the robot. A solution is to start learning in simulation before transferring the policies to real robots [38][39]. With this approach, it

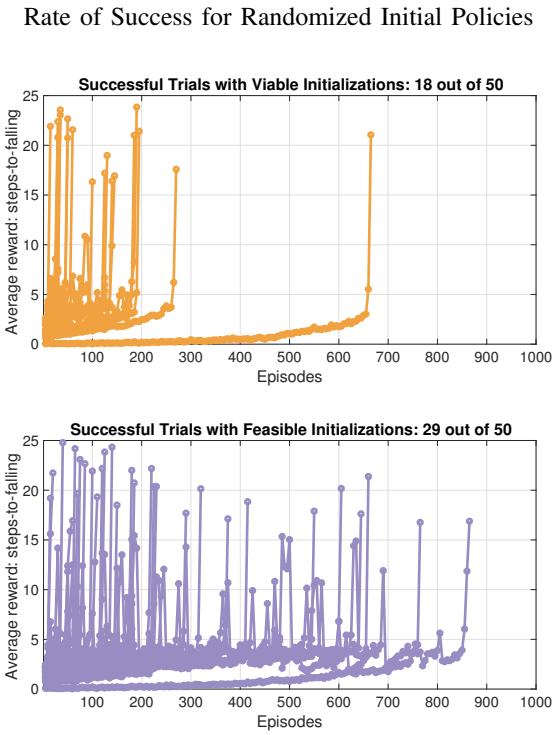


Fig. 5: Fifty initial policy parameters were tested, sampled at random with a minimum chance of taking a 0.5 steps on average when using a viable initialization. For visual clarity, only successful trials are plotted.

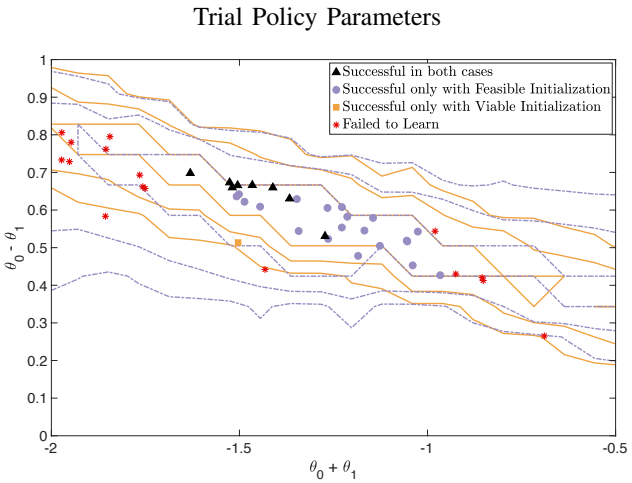


Fig. 6: Examining the location of the initial policy parameters on the reward landscape shows just how brittle the viable initializations are. The distribution of the parameters which fail for both initializations also suggests some further structure we have not explored here. The landscape contours with viable initializations (solid orange line) and feasible initializations (dashed purple line) are overlaid for reference.

is important not to mirror the same conditions of the robot in simulation, but purposefully explore with more aggressive and potentially unviable initializations.

While this approach is promising, a simulation still relies on a model even if the policy is learned model-free. One of the attractions of model-free learning is the reduced need for expert knowledge and engineering effort in order to directly deploy robots in the field. To this end, our results suggest a different emphasis for robot design is needed. A robot should not only be mechanically sturdy so it can survive failures, but there should be meaningful actions to be explored in unviable states. As we show with the SLIP model (see Fig. 2), the choice to fall immediately or to stumble several steps before falling is what allows the system to learn from unviable states. This is a property of the system dynamics, and therefore the hardware design and not the controller design. Especially when building legged robots that try to mimic a SLIP-like behavior [27][28], these aspects should be considered in addition to measures such as passive stability and energy efficiency.

References

[1] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of go without human knowledge", *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 1. MIT press Cambridge, 1998, vol. 1.

[3] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[4] E. Heijmink, A. Radulescu, B. Ponton, *et al.*, "Learning optimal gait parameters and impedance profiles for legged locomotion", in *Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on*, IEEE, 2017, pp. 339–346.

[5] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion", in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, IEEE, vol. 3, 2004, pp. 2619–2624.

[6] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search", in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 156–163.

[7] N. Chentanez, A. G. Barto, and S. P. Singh, "Intrinsically motivated reinforcement learning", in *Advances in neural information processing systems*, 2005, pp. 1281–1288.

[8] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning", in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 1.

[9] M. Kalakrishnan, P. Pastor, L. Righetti, *et al.*, "Learning objective functions for manipulation", in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 1331–1336.

[10] Y. Bengio, J. Louradour, R. Collobert, *et al.*, "Curriculum learning", in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 41–48.

[11] A. Karpathy and M. Van De Panne, "Curriculum learning for motor skills", *Advances in Artificial Intelligence*, pp. 325–330, 2012.

[12] J. Randlov and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping", in *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, pp. 463–471.

[13] S. Heim, F. Ruppert, A. A. Sarvestani, *et al.*, "Shaping in practice: Training wheels to learn fast hopping directly

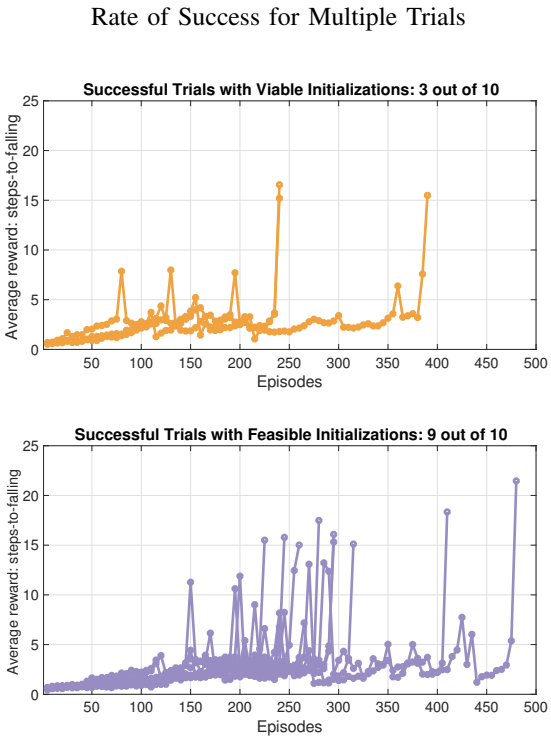


Fig. 7: Ten trials starting from the same initial policy parameters were repeated, with viable initializations (top) and feasible initializations (bottom). The improved reliability with feasible initializations is clear, with 9 out of 10 trials succeeding, compared to 3 out of 10 for the viable initializations. For visual clarity, only successful trials are plotted.

in hardware", *arXiv preprint arXiv:1709.10273*, 2017, Accepted to Robotics and Automation (ICRA), 2018 IEEE International Conference on.

[14] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates", in *International Workshop on Hybrid Systems: Computation and Control*, Springer, 2004, pp. 477–492.

[15] N. Smit-Anseeuw, R. Vasudevan, and C. D. Remy, "Safe online learning using barrier functions", *Proceedings of Dynamic Walking*, 2017.

[16] J.-P. Aubin, *Viability theory*. Springer Science & Business Media, 2009.

[17] R. Blickhan, "The spring-mass model for running and hopping", *Journal of biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, 1989.

[18] R. J. Full and D. E. Koditschek, "Templates and anchors: Neuromechanical hypotheses of legged locomotion on land", *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, 1999.

[19] H.-M. Maus, S. Revzen, J. Guckenheimer, *et al.*, "Constructing predictive models of human running", *Journal of The Royal Society Interface*, vol. 12, no. 103, p. 20 140 899, 2015.

[20] T. Cnops, Z. Gan, and C. D. Remy, "The basin of attraction for running robots: Fractals, multistep trajectories, and the choice of control", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 1586–1591.

[21] G. Piovan and K. Byl, "Reachability-based control for the active slip model", *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 270–287, 2015.

[22] İ. Uyanık, U. Saranlı, and Ö. Morgül, "Adaptive control of a spring-mass hopper", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 2138–2143.

[23] A. Wu and H. Geyer, "The 3-d spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments", *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1114–1124, 2013.

[24] W. C. Martin, A. Wu, and H. Geyer, "Experimental evaluation of deadbeat running on the atrias biped", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1085–1092, 2017.

[25] S. Rezazadeh, C. Hubicki, M. Jones, *et al.*, "Spring-mass walking with atrias in 3d: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot", in *Proceedings of the ASME 2015 Dynamic Systems and Control Conference*, 2015.

[26] P. Terry, G. Piovan, and K. Byl, "Towards precise control of hoppers: Using high order partial feedback linearization to control the hopping robot frank", in *Decision and Control (CDC), 2016 IEEE 55th Conference on*, IEEE, 2016, pp. 6669–6675.

[27] C. Hubicki, J. Grimes, M. Jones, *et al.*, "Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot", *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1497–1521, 2016.

[28] D. Lakatos, W. Friedl, and A. Albu-Schäffer, "Eigenmodes of nonlinear dynamics: Definition, existence, and embodiment into legged robots with elastic elements", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1062–1069, 2017.

[29] M. Hutter, C. D. Remy, M. A. Höpflinger, *et al.*, "Slip running with an articulated robotic leg", in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 4934–4939.

[30] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[31] A. Seyfarth, H. Geyer, M. Günther, *et al.*, "A movement criterion for running", *Journal of biomechanics*, vol. 35, no. 5, pp. 649–655, 2002.

[32] J. Rummel and A. Seyfarth, "Stable running with segmented legs", *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 919–934, 2008.

[33] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, *et al.*, *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2007, ch. 4.

[34] L. R. Palmer III and C. E. Eaton, "Periodic spring–mass running over uneven terrain through feedforward control of landing conditions", *Bioinspiration & biomimetics*, vol. 9, no. 3, p. 036 018, 2014.

[35] G. Piovan and K. Byl, "Two-element control for the active slip model", in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 5656–5662.

[36] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning", *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[37] R. S. Sutton, D. A. McAllester, S. P. Singh, *et al.*, "Policy gradient methods for reinforcement learning with function approximation", in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[38] X. B. Peng, M. Andrychowicz, W. Zaremba, *et al.*, "Sim-to-real transfer of robotic control with dynamics randomization", *arXiv preprint arXiv:1710.06537*, 2017.

[39] W. Yu, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification", *arXiv preprint arXiv:1702.02453*, 2017.