Enabling Autonomous Legged Robot Agility

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Mechanical Engineering

Joseph C. Norby

B.S. Mechanical Engineering, University of Notre Dame

Carnegie Mellon University
Pittsburgh, PA

August, 2022

Acknowledgements

Firstly I would like to thank Carnegie Mellon University for providing me the opportunity to conduct research in a world-class environment, and all the faculty and staff who have supported me during my time in Scaife and Wean Halls (Prof. Mark Bedillion, Chris Hertz, Ed Wojciechowski, and John Fulmer in particular). I am also very grateful for the support of other institutions who have made my research possible: the National Science Foundation (grants DGE-1745016, IIS-1704256, ECCS-1924723, and CCF-203085), Chevron CTC, and the Army Research Office (grant W911NF-19-1-0080).

I would like to thank the members of my committee for guiding me through the forays of graduate studies: Prof. Maxim Likhachev for sparking my interest in motion planning and encouraging me to focus on theory in addition to implementation, Prof. Chris Atkeson for laughing at my simple ideas and motivating me to think bigger, and Prof. Jonathan Hurst for pushing me to be less concerned with uninteresting details and more focused on good science. Perhaps the most thanks of all go to my committee chair and research advisor, Prof. Aaron Johnson. Without your caring and thoughtful guidance both in research and life I would be very far from where I am today. Thank you for taking a chance on me in your first Ph.D. cohort, and I look forward to hearing about exciting new projects at many Mad Mexes in the future.

Huge thanks are in order for all the past and present members of the Robomechanics Lab at CMU. To fellow critter and co-Old Boar Catherine, it has been immensely fun building the lab together and I think we've done pretty well at creating an environment which doesn't take itself too seriously. Thank you to Ankit, Joe, Nathan, Sean, James, Paul, and Stutt for making every day in lab truly enjoyable. To all the members of the Quad-SDK team (Yanhao, Ardalan, Jiming, Justin, Stutt again, Qishun, and Nikolai), I'm very proud of the work we've done and honored to have done it alongside you. Special thanks in particular go to Ardalan for helping get the last few pushes out of the door, and Yanhao for putting up with two years of my mentorship and working deep in the trenches getting the darn robot to do what it is supposed to.

I must also thank my wonderful friends and family for supporting my journey from weeknights

in middle school making barely functional battle bots to creating this document. To Abel for enthusiastically answering my never-ending questions about ROS. To Jonathan, Patrick, and Charlie for keeping me focused on the more entertaining side of the big picture (also for inspiring the names of two of our robots). To my siblings Greg, Chris, and Ellie (and the outlaws!) for always being just a phone call away from love and assurance. To my parents Lynn and Kevin for their unwavering support for my adventures even if they've taken me far from home. You've played a bigger role in creating this document than you know. Lastly, thank you to my darling partner-in-crime Nicole (and our cat Syl) for always believing in me even when I didn't. I hope you know how much you inspire me, and I cannot wait to start the next part of our journey together. Let's go exploring!

Abstract

Successful deployment of legged robots in industrial applications such as environmental monitoring, material handling, or inspection requires that platforms perform complicated tasks quickly in unstructured environments. Unlocking the capabilities offered by legged morphologies requires autonomously harnessing their capability for agility to perform these tasks. However, autonomous and agile legged robot locomotion over unstructured terrain remains difficult due to underactuation, the hybrid nature of intermittent contact, as well as kinematic, dynamic, and computational constraints. Overcoming these restrictions requires some level of reasoning about how best to navigate the environment, yet often it is not clear what level of reasoning is best to solve a task. Simple models enable efficient computation and are generally robust to many forms of error, but they cannot capture all the salient features of a complex, multi-behavioral system such as a legged robot. Highly detailed models may be able to capture these features, but often at the expense of physical and computational robustness.

This thesis seeks to overcome this apparent trade-off and enable autonomous legged robot agility by adapting the complexity of the solution to the task at hand. These methods include a global motion planner with mixed-complexity motion primitives and an accompanying open-source full stack software framework for deployment on agile quadrupeds, a novel form of model predictive control that adapts its model to ensure feasibility and improve efficiency, and developments in bio-inspired robotic tail design to improve stability while reducing overhead costs.

The novel global motion planner enables autonomous agility by constructing long-horizon plans in real-time. It employs a mixture of motion primitives of varying fidelity to navigate from the current position to the goal while avoiding (or leaping over) obstacles and uneven terrain. This planner is shown to compute nearly an order of magnitude faster than comparable algorithms over dynamically challenging environments.

This global planner is featured within an open-source software package – Quad-SDK – which enables other researchers to deploy agile autonomy to their quadrupedal platforms. The package implements a full planning and control hierarchy which also includes a nonlinear model predictive

controller with a novel warm starting technique that enables highly stable execution of dynamic motion plans. Several experiments are shown in simulation and hardware which demonstrate the system's ability both plan and execute long horizon motion plans which include leaps. The package is also well supported with software tools to enable rapid development.

Even with efficient global planners, employing reduced-order models will generally result in locomotion which encounters constraints in the full-order representation of the system. This work presents a form of model predictive control which adapts the complexity of the model to capture the salient dynamics and constraints of the task. It is shown that under certain well-known template and anchor conditions this method yields provable stability properties, and enables simplification of the optimal control problem which results in improved performance. This method is benchmarked against fixed complexity formulations over candidate dynamic behaviors and is shown to be more stable than reduced-order configurations and more efficient than full-order configurations.

The novel tail design explores aerodynamic drag as a tool to regulate angular momentum independent of foot contact for improved agility. A new metric for the effectiveness of aerodynamic drag is presented and analyzed. Comparison to standard inertial effectiveness shows that aerodynamic drag tails can perform the same reorientation tasks as inertial tails but for a fraction of the inertia. The utility of these tails is demonstrated in hardware for disturbance rejection and locomotion assistance tasks, showcasing their enhanced practicality over inertial tails in providing leg-independent momentum regulation.

Contents

List of Tables

List of Figures

1 Introduction

1.1 The Path to Useful Legged Robots

The ability of mobile robots to combine computational tools and automation with the capability to move around in and interact with their environment enables a wide variety of both economically and socially driven tasks. Many of these tasks, such as environmental monitoring, inspection of dangerous areas, delivery, and space exploration, occur in unstructured terrain like those shown in Figure 1. These are environments that cannot be easily altered to improve the mobility of the robot and often include obstacles which the robot must navigate. Of the different categories of ground mobile robots – primarily wheeled, tracked, and legged – legged robots have a unique potential for mobility in unstructured environments due to the diversity of possible interactions between the robot and the environment. The ability to apply an array of forces at different contact locations enables behaviors like stepping, turning, jumping, and climbing to overcome obstacles and thus improve mobility.

Despite the potential for mobility in difficult terrain, legged platforms are rarely deployed for any industrial applications of mobile robots. Tracked inspection robots provide data and imaging in situations dangerous for humans [Robohub, 2015], wheeled autonomous vehicles show great promise in improving highway safety and assisting ground delivery [Litman, 2017], wheeled robots even currently explore other planets [NASA, 2020], yet legged robots lag behind these counterparts in industrial usage. Some promising platforms are under development such as Spot for inspection tasks [Boston Dynamics, 2020], Digit for package handling and delivery [Agility Robotics, 2020], and Vision 60 for surveillance applications [Ghost Robotics, 2020], but these platforms are still confined to walking over relatively structured terrain and often rely on the guidance of a user or the tracking of pre-determined paths. The potential for legged robot mobility in unstructured terrain is unrealized due to two main limitations: agility and autonomy. First, a practical mobile robot must be agile in order to accomplish tasks quickly and efficiently, and to enable the navigation of kinematically challenging terrain such as stairs, steep slopes, or large obstacles. Current robotic

Figure 1: Legged robots must navigate unstructured environments such as those shown here to perform useful tasks. These environments often require the ability to step or leap to overcome obstacles. Current legged robots struggle to autonomously perform these behaviors. Applications of interest shown clockwise from top-left: space exploration (image credit: NASA), outdoor environmental monitoring (image credit: www.hiketricities.com), urban delivery or mapping (image credit: author), and search and rescue (image credit: Linda Davidson/The Washington Post)

platforms are much slower than humans at navigating such terrain. Second, mobile robots must also be intelligent enough to autonomously interact with many environments in the right way to enable navigation. Current platforms often make assumptions about terrain structure or restrict themselves to a small subset of simple motions. They also often require instruction from human operators or direct intervention in the event of failures caused by uncertainty or disturbances.

1.2 Agility and Autonomy

Before exploring methods to improve agility and autonomy, some definitions are in order to clarify the usage of these terms. Many different agility metrics have been proposed in robotic, biological,

and even athletic literature as a way to quantify the locomotion capabilities of a system although none definitely and comprehensively quantify agility. Some local metrics quantify the ability of a system to maneuver over short time horizons such as acceleration, Froude number, specific power, or specific agility [Alexander, 1984; Duperret et al., 2016; Roberts et al., 2011]. Others take a more global view by quantifying the performance of longer horizon maneuvers such as turning radius, time to complete particular motions, or combinations of a number of benchmarks [Eckert and Ijspeert, 2019; Sheppard and Young, 2006]. Since this work discusses agility in the context of completing navigation tasks (i.e. moving from point A to point B), agility here refers to the completion of these tasks with high kinetic energy. This translates to the ability to perform acrobatic motions such as leaping or rapid turning, but also has a direct correlation to desirable performance in industrial applications, as higher kinetic energy corresponds to completing tasks more quickly. While no one metric entirely captures this notion, average kinetic energy while navigating from one state to another is a useful starting point.

Autonomy is even more difficult to quantify than agility – there are no SI units that could reasonably express such an abstract notion. Prior literature has sought to qualitatively define "autonomy levels" such as the NIST Autonomy Levels for Unmanned Systems [Huang et al., 2005], the SAE Levels of Driving Automation [International, 2014], or other qualitative definitions [Beer et al., 2014]. Other works have sought to quantify autonomy through a series of simulation-derived performance metrics based on properties like environment complexity, available information, and system performance [Lampe and Chatila, 2006]. In this work, autonomy refers to the ability of a system to successfully complete locomotion tasks with minimal information or intervention from an external operator. This encourages autonomous path planning rather than teleoperation, but also requires a high level of robustness to uncertainty, disturbances, or changes in the environment as the inability to reliably overcome these challenges would require operator intervention to complete the task. Thus metrics such as planning time, success rate, and environment complexity can together shed light on the autonomy of a system.

1.3 Challenges in Legged Locomotion

It is important to note that the practical shortcomings of current platforms are generally not due to limitations of hardware components. The force production of a modern motor can be comparable to that of human muscle [Marden and Allen, 2002], and current battery technology is sufficient to power legged robots for a few hours at a time. Many existing legged robots have shown impressive agility, from running faster than humans [IEEE Spectrum, 2012] to jumping almost 30 times their body height [Kovac et al., 2008] to performing backflips as early as 1988 [Hodgins and Raibert, 1988]. The primary reason that legged robots remain slow and not autonomous in unstructured terrain lies in the lack of proper algorithms to autonomously plan and execute challenging navigation tasks while accounting for this terrain.

Constructing these algorithms remains difficult due to the nature of agile legged locomotion. Successful locomotion requires careful regulation of both linear and angular momentum to ensure the robot has just the right velocity to leap onto a ledge or to stabilize after a foot slips unexpectedly. Constructing plans to precisely regulate momentum is challenging because both the kinematics of locomotion and the presence of intermittent contact render the equations that describe the motion of these systems both nonlinear and discontinuous. These features complicate the assumptions of many motion planning and control methods. Additionally, actuation and kinematic limits subject legged robots to underactuation, meaning they cannot achieve arbitrary accelerations to complete a given task [Tedrake, 2009]. This is particularly evident in flight phases – when the robot has no legs in contact with the ground, it has almost no control authority and is at the whim of ballistic motion. Together these effects – illustrated in Figure 2 – restrict both the ability and rate at which a robot can alter its own momentum. Reliably attaining agile and autonomous locomotion in unstructured environments thus requires methods that allow a robot to plan and regulate momentum despite these limitations.

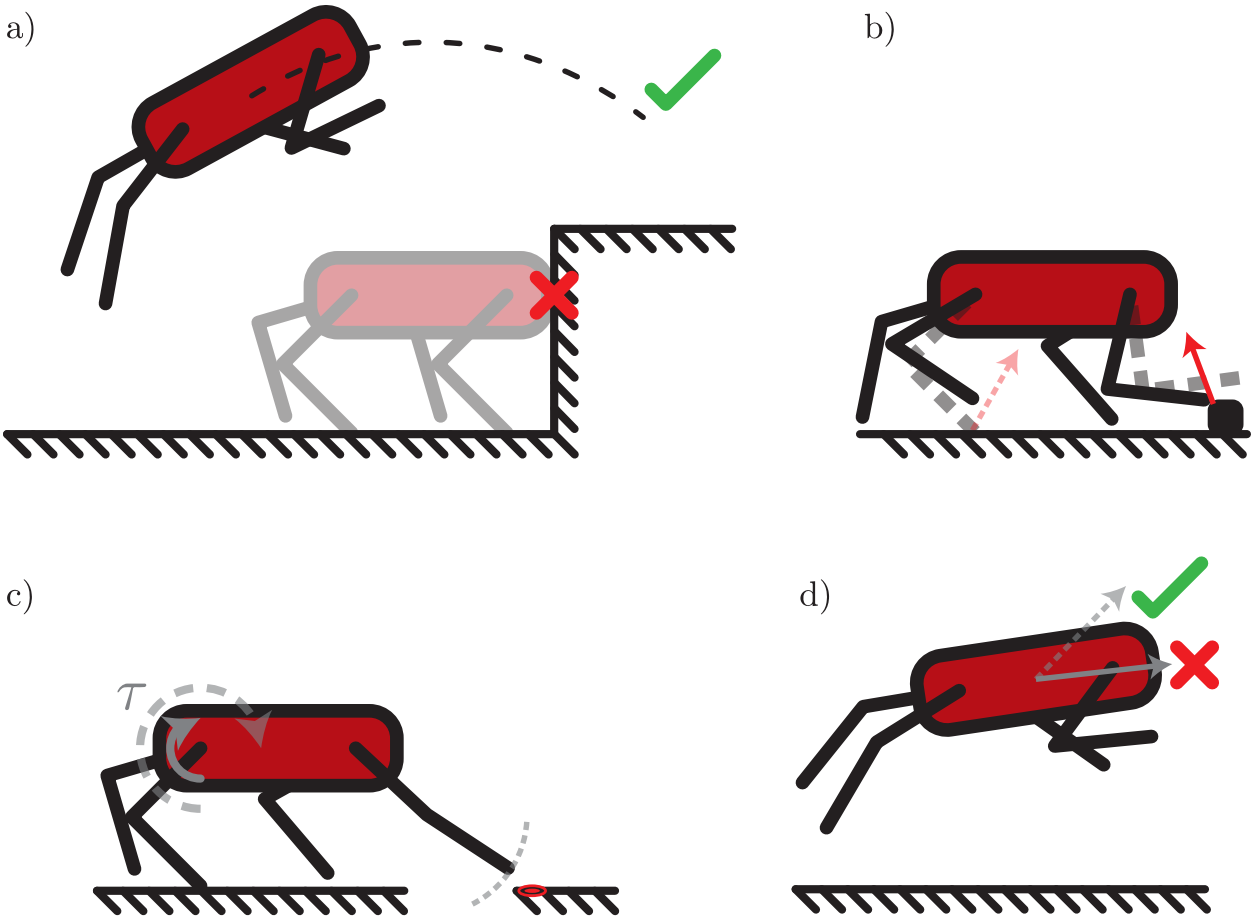


Figure 2: Agile and autonomous legged robot locomotion remains difficult for a number of reasons, including a) the need to plan ahead, b) errors in desired ground reaction forces caused by unexpected changes in contact, c) kinematic and dynamic limits, and d) underactuation which restricts the ability to regulate momentum (particularly when ground contact is sparse). In each figure, dashed lines indicate desired properties while solid lines indicate actual properties.

1.4 Locomotion Complexity and Task Difficulty

Researchers have developed a number of strategies for planning and executing locomotion tasks in the presence of these challenges, and these methods can be broadly differentiated by complexity of the system and the amount of data they employ to solve the task as illustrated in Fig. 3. "Complex" systems attempt to process large amounts of data with a high fidelity representation of the system state, dynamics, and constraints. These types of methods include high-dimensional trajectory optimization, search-based planning, dynamic programming, or end-to-end reinforcement learning. These techniques have produced highly intelligent systems capable of performing difficult tasks

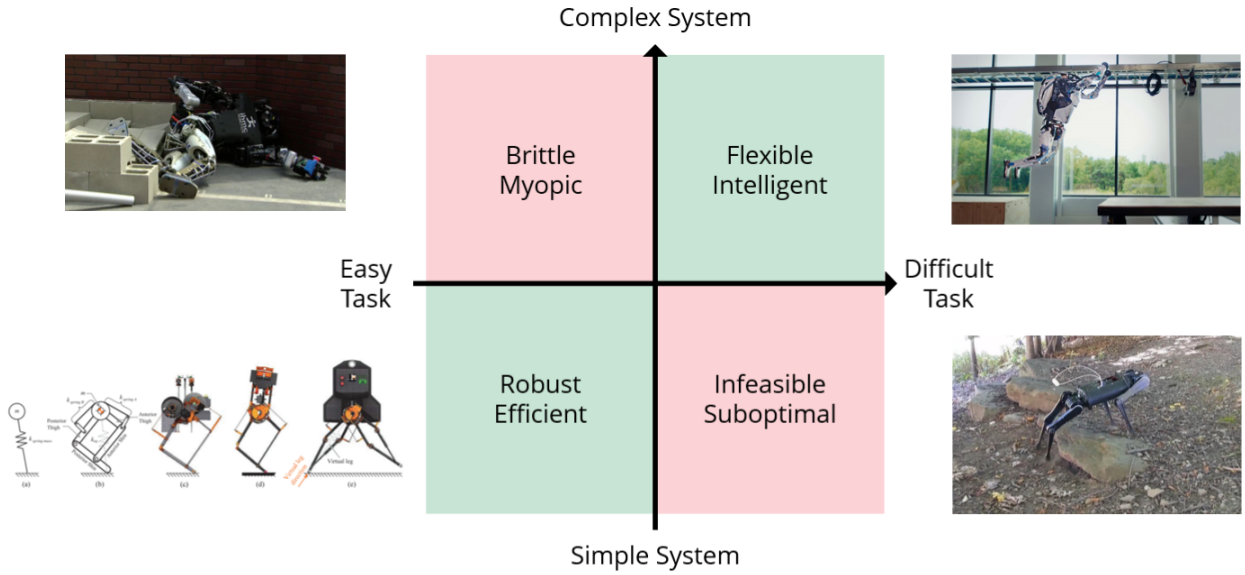


Figure 3: The complexity of a locomotion system should match the difficulty of the task to maximize the likelihood of success (image credits, clockwise from top-left: IEEE Spectrum, Boston Dynamics, author, [Hubicki et al., 2016]).

such as coordinated whole-body control, generalization to other tasks such as manipulation, or end-to-end learning approaches that can operate in a wide range of conditions [Dai et al., 2014; Kuindersma et al., 2016; Kumar et al., 2021; Lee et al., 2020]. These systems typically require as much information about the task at hand as possible through diverse sensor arrays, advanced state estimation algorithms, or huge amounts of experiential data. This results in susceptibility to modeling errors, sensor inaccuracy, computational limits, and local minima, which can yield brittle or myopic performance even when performing easier tasks.

"Simple" systems eschew these details for lower-dimensional representations which attempt to distill the task into its most fundamental components, which can then be resolved in a much more straightforward manner. Common tools employed by these simpler systems are model reductions and hierarchies, heuristics, passive dynamics, or learned latent representations of salient properties. Prior work has demonstrated impressive robustness and efficiency for simple tasks such as steady-state locomotion over flat or mildly rough terrain [Bhounsule et al.; Collins et al., 2001; Hubicki et al., 2016; Kajita et al., 2003; Raibert, 1986]. These systems typically function best when applied to a particular task that can effectively be decomposed into simpler subtasks or when large amounts

of data processing are unavailable. The drawback of these approaches is that by definition they ignore certain details of the highly-complex system, and these omissions can often yield solutions that are at best sub-optimal and at worst completely infeasible.

While researchers often espouse one of these solution methods over the other based on the task they are solving, in truth many tasks consist of a range of difficulties. Consider a legged system walking in unstructured terrain such as those in Fig. 1. Many environments contain large portions of relatively flat terrain segmented by regions of larger elevation changes or roughness. A system with a high degree of complexity reasoning over a short horizon may fail to recognize that its greedy actions are directing it to a local minima when it should be accelerating to prepare for some agile behavior. Likewise a system reasoning about simplified dynamics may not recognize that an upcoming constraint in its high-dimensional space is about to cause a catastrophic failure.

This trade-off could be resolved by matching the complexity of the solution to the difficulty of the task. If the system were to leverage a simple model in areas where it accurately captured the dynamics and constraints of the task, it could spend more of its resources synthesizing behaviors where these assumptions break down. Likewise by reducing the required complexity in simple portions the system can improve robustness to uncertainty. This concept of dividing cognitive effort based on the task at hand is well-studied in fields such as behavioral economics [Kahneman, 2011], and has already been applied to some robotic motion planning tasks [Fridovich-Keil et al., 2018; Gochev et al., 2012; Styler and Simmons, 2017].

This methodology can apply to mechanical systems as well as data-processing systems. Increasing the complexity of a legged platform by adding additional appendages could expand the range of behaviors the system could achieve, at the cost of degraded performance during behaviors where the additional complexity is not needed. Such a trade-off could be avoided through designs that adapt to the task at hand, similar to how the fins of aquatic animals expand or retract as needed for propulsion [Alexander, 2013]. In the context of legged locomotion, this could amount to systems that improve control authority of the system when terrestrial propulsion is absent or unreliable, and minimize their added effects otherwise to maintain efficiency and robustness.

1.5 Dissertation Overview

This work aims to enable autonomous legged robot agility through several approaches that match the complexity of the system to the locomotion task difficulty. Section 2 presents a novel global motion planning algorithm that efficiently finds collision-free trajectories for a reduced-order model of a legged platform [Norby and Johnson, 2020]. This algorithm employs motion primitives of differing levels of complexity to enable both long-horizon locomotion over simple terrain as well as leaping behaviors over rough terrain. Experiments over several example terrains show that this approach can solve a wide range of navigation tasks and is faster than comparable methods.

Section 3 describes the integration of this global planning algorithm into a hierarchical planning and control framework – Quad-SDK – capable of performing highly agile behaviors autonomously [Norby et al., 2022b]. This framework has been released as an open-source software package to enable other researchers to implement such capabilities within their own applications. This section discusses how these components interact as well as additional implementation details required for agile locomotion. Results are presented over a range of environments in simulation and hardware demonstrating the ability of the system to both plan and execute agile behaviors.

However, executing reduced-order motion plans on the edge of constraint boundaries requires methods that can resolve infeasibilities without requiring overly conservative approximations. Section 4 presents a formulation of model predictive control (MPC) which adaptively reasons about the complexity of the model required for feasibility and stability [Norby et al., 2022a]. The method solves MPC problems with a simple model for dynamics and constraints over regions of the horizon where such a model is feasible and a complex model where it is not. It leverages an interleaving of planning and execution to iteratively identify these regions, which are described by well-studied template and anchor relations [Full and Koditschek, 1999]. It is shown that this method provides formal stability and feasibility guarantees and yields a larger region of attraction than directly reasoning over the complex system. This section presents experiments in simulation and hardware for a quadrupedal robot executing agile behaviors over rough terrain which show that this adaptive framework can enable faster solve times and more stable motion than fixed-complexity implemen-

tations.

Even with these contributions, locomotion is still limited by the presence and quality of contact with the terrain, and methods that introduce complexity through additional actuators such as tails often result in excessive payload costs when these actuators are idle. Section 5 studies the use of tails that leverage aerodynamic drag to reorient the system when needed but are otherwise much more lightweight [Norby et al., 2021]. We present a model of the aerodynamic drag and from this derive a metric that allows for direct comparison between aerodynamic and inertial tails. Motivated by this model, we construct a tail to maximize this effectiveness while minimizing inertia. We demonstrate the utility of this tail for two dynamic behaviors executed on a quadrupedal robot. First, in aerial reorientation the robot achieves a 90 degree rotation within one body length of fall at the same performance as an inertial tail but with just 37% of the normalized inertia. Second, the forward acceleration of the robot is improved by 12% despite increasing the system mass by 10% over a tailless version.

Section 6 summarizes the overall contributions and highlights key remaining challenges and opportunities towards attaining autonomous legged robot agility.

2 Fast Global Motion Planning for Dynamic Legged Robots

2.1 Introduction

As discussed in Section 1, automating navigation tasks requires the ability to plan ahead to ensure the robot has the appropriate positioning and velocity to execute the desired motion. This planning is challenging due to nonlinear dynamics, underactuation, intermittent contact including flight phases, and dependence on the terrain itself.

Model-based trajectory planning has proven an effective tool for planning motions while accounting for these challenges. The most straightforward form of trajectory planning is to give the planner full knowledge of the forces the robot can apply to the world and the constrained manner in which it can apply them, enabling the planner to predict how it can feasibly navigate the terrain and generate corresponding control inputs [Mombaur, 2009]. These methods have demonstrated impressive results in simulation, yet their numerical complexity renders them infeasible for real-time hardware deployment.

Rather than solve for the entire motion all at once with a high fidelity model, many successful robot implementations employ hierarchical control structures [Fankhauser et al., 2018; Kuindersma et al., 2016; Zucker et al., 2011]. Such an architecture breaks the problem into multiple subproblems that can be solved in parallel. Typically these layers include a long-horizon "global" planner to determine a rough path through the terrain, a short-horizon "local" planner that refines this motion and often selects footholds, and a low-level controller that computes joint torques to send to the motors. Distributing model complexity – with higher complexity for shorter horizons – ensures that each layer can be computed in real-time, yet presents the challenge of ensuring that any model simplifications can be resolved by lower layers.

The global planner is critical to this pipeline due to its place at the top. A good global planner must have an accurate idea of what the robot can and cannot do to ensure that the other layers can resolve the motion, but also must have a long enough horizon to avoid local minima which is difficult to achieve in real time. Most global planners used in existing hierarchies achieve fast

Figure 4: Demonstration of the proposed algorithm generating a long dynamical plan over terrain challenging for a legged robot in under three seconds. The trajectory from the start pose to the goal (green circle) is overlaid on an image of the environment used to generate the plan.

solution times by either ignoring dynamics to employ geometric planning methods like A* [Hornung et al., 2012], restricting the horizon to only a few steps [Winkler et al., 2018a], or relying on traversability maps that use simple heuristics like maximum step height to avoid certain areas of the terrain [Wermelinger et al., 2016].

This work introduces a fast global motion planning framework that can compute long-horizon feasible robot pose trajectories (body position and orientation) from which whole-body motions can be found, while accounting for intermittent contact, underactuation, and constrained kinematics and dynamics [Norby and Johnson, 2020]. The framework is based on a computationally efficient state and action parameterization that is effective at exploring environments dynamically while also permitting the connection of two states for fast exploitation of trivial terrain. The framework also supports path length reduction methods to produce high quality paths, which is important

for practical robot implementation. We present the results of four simulation experiments: the first demonstrates the validity of the constraint approximations with a whole-body trajectory optimization, the second analyzes the speed and horizon length of the planner on several terrains, the third demonstrates path length reduction capabilities, and the fourth shows that this planner finds paths faster than state-of-the-art legged motion planning algorithms on a benchmarking environment.

2.2 Related Work

Legged robots have long shown remarkable ability to perform dynamic behaviors, demonstrating walking, running, and even front-flipping as early as the 1980s [Hodgins and Raibert, 1988; Raibert, 1986; Zeglin, 1991]. Though these robots exhibit surprisingly robust locomotion, they rely on hand-tuned motions or clever mechanical stability through compliant legs or tails. These simple methods (which form the basis for the proposed algorithm) are adept at walking over relatively flat terrain but cannot easily handle obstacles that require leaping. More recent robots have demonstrated impressive obstacle-leaping behaviors [Haldane et al., 2016; Johnson and Koditschek, 2013b; Park et al., 2015], but without a global planner that includes flight phases they cannot autonomously determine when and how to perform these tasks in unstructured terrain.

Accounting for unstructured terrain requires a notion of the capabilities of the system, and a method to employ this knowledge to plan a path through the environment. The clearest way to achieve this is through trajectory optimization with a full-order representation of both the kinematics and dynamics of the robot, an accurate model of the terrain, and a nonlinear solver that can find a feasible trajectory from the start to the goal. This method has shown impressive results in walking, running, jumping, and object manipulation [Mombaur, 2009; Mordatch et al., 2012; Posa et al., 2014], yet fully representing the nonlinearities and intermittent contact of legged locomotion yields extremely large and highly constrained nonlinear programs that take hours to solve on state-of-the-art solvers. Researchers have shown that ignoring any limb dynamics and modeling just the centroidal momentum of the system is sufficient to perform challenging tasks like running, jumping, and brachiation [Dai et al., 2014; Kuindersma et al., 2016; Orin et al., 2013]. This ap-

proach reduces computation times from hours to minutes, but the problem still cannot be solved fast enough to be used in real time.

These solution times can be improved by further reducing the model complexity while relying on the aforementioned hierarchical control structure to handle any small feasibility violations. One common method of reducing model complexity is approximating kinematic constraints with more efficiently computed heuristics such as bounding boxes [Caron et al., 2016; Winkler et al., 2018a], reachability sets [Shapiro et al., 2005; Tonneau et al., 2018], learned measures [Carpentier and Mansard, 2018; Kalakrishnan et al., 2009], or ignoring them entirely [Audren et al., 2014; Di Carlo et al., 2018]. Conservative kinematic approximations reduce the space of trajectories, but overly optimistic approximations can lead to infeasible plans that cannot be resolved lower in the hierarchy. Other methods simplify the dynamics further by enforcing quasi-static, inverted pendulum, or Zero Moment Point dynamics [Bartoszyk et al., 2017; Fankhauser et al., 2018; Hauser et al., 2008; Hornung et al., 2012; Zucker et al., 2011]. These reductions allow for planning over a long horizon of many steps, but restrict the dynamics of the planned motion to trivial applications.

Some planners augment the terrain map with a notion of traversability, such that regions of the environment are processed and labeled with regards to the capabilities of the robot [Brunner et al., 2013; Wermelinger et al., 2016]. By conducting simple tests of a walking controller's performance based on factors such as terrain slope, surface roughness, step height, and footprint size, features of a terrain can be given a score based on the likelihood of successful navigation. Then, common sampling-based methods such as RRT* [Karaman and Frazzoli, 2011] can be employed to maximize the traversability of the given path. This approach has some features of a dynamic global planner in that it can be computed in real-time and account for the dynamic capabilities of the robot, but the construction of the traversability map is highly heuristic, robot and controller dependent, and unable to synthesize new motions to navigate.

One of the most promising methods, [Fernbach et al., 2017], utilizes the RRT-Connect algorithm [Kuffner and LaValle, 2000] to synthesize kinodynamic motions for legged locomotion over long horizons in under a minute. This algorithm connects states by solving linear programs (LPs) to

determine acceleration bounds and valid footstep locations, then interpolates between states while using these bounds to check for feasibility. The algorithm plans for flight phases by identifying states that do not satisfy a reachability requirement and trying to find feasible ballistic trajectories to connect to the path to surpass them. The utilization of RRT-Connect allows this method to plan over longer horizons than other motion planners [Deits and Tedrake, 2014; Mordatch et al., 2012], but the repeated solving of LPs and heuristic approach to handling flight phases restrict the horizon under which this method can plan. The proposed algorithm in this work parameterizes the state and action spaces similarly for compatibility with RRT-Connect, but in a way that allows for the automatic generation of feasible actions by randomly sampling within bounds on force constraints. This parameterization also directly incorporates flight phases rather than identifying them with heuristics. In addition, the proposed algorithm incorporates path length reduction methods such as RRT* and anytime planning with short-cutting to produce high quality paths more amenable to execution on a robot.

2.3 Planning Algorithm

The core of the proposed algorithm is the reduction of the dynamics of legged locomotion to a state-action pair framework conducive to both RRT-Connect for rapid exploration of mostly flat terrain and kinodynamic RRT (KD-RRT) for exploration of challenging terrain [LaValle and Kuffner Jr, 2001]. This is achieved by reducing the system state to just the robot's body position, orientation, and their velocities, with double integrator dynamics to determine the overall path through the environment. So long as the resulting trajectories can be feasibly tracked with whole-body motions – as shown in Section 2.4.1 – this reduction permits efficient long horizon planning while allowing more complex tasks such as footstep planning to be solved with existing shorter-horizon methods such as [Winkler et al., 2018a; Zhao and Sentis, 2012].

Pose motion is expressed as a sequence of piece-wise polynomials due to their computational efficiency and closed form solutions. Expressing motion with polynomials also allows dynamic constraints such as ballistic motion during flight phases and non-adhesive forces, friction cones,

and actuation limits in stance phases to be automatically satisfied through the selection of appropriate polynomial coefficients. Motions are then planned by stitching these piece-wise trajectories in a way that satisfies kinematic constraints, in particular collision avoidance and reachability.

2.3.1 State Parameterization

Sampling-based planning methods rely on a set of states and actions that map to other states. The algorithm proposed here defines these discrete time states as the position, orientation, and velocities of the base of the robot. This sort of COM trajectory planning is not itself a new concept [Klemm et al., 2015; LaValle and Kuffner Jr, 2001], but here we explicitly reason about intermittent contact and formulate the actions to account for the dynamics and kinematics of the system.

The discrete time states are defined at the beginning of a stance phase and denoted as,

$$\begin{aligned} \mathbf{s}_i &= \begin{bmatrix} \mathbf{q}_i \\ \dot{\mathbf{q}}_i \end{bmatrix} \\ &= \begin{bmatrix} q_{x,i} & q_{y,i} & q_{z,i} & q_{p,i} & \dot{q}_{x,i} & \dot{q}_{y,i} & \dot{q}_{z,i} & \dot{q}_{p,i} \end{bmatrix}^T, \end{aligned} \tag{1}$$

where $\mathbf{s}_i \in \mathbb{R}^8$ is the state at the beginning of the ith stance phase, $\mathbf{q}_i \in \mathbb{R}^4$ is the position and pitch of the system, x, y, z subscripts denote Cartesian directions, and the p subscript denotes the pitch. To reduce the dimensionality of the state space, roll is set to zero and yaw is defined as $\texttt{atan2}(\dot{q}_{y,i}, \dot{q}_{x,i})$ so that the robot's heading aligns with its velocity. A stance phase is defined as a period of motion where a ground reaction force (GRF) is applied to the robot (and may correspond to multiple overlapping stance phases for each individual foot). Each stance phase is then followed by a flight phase consisting of ballistic motion, after which the next state $\mathbf{s}_{i+1}$ denotes the beginning of the following stance phase. Figure 5 shows example states and the stance and flight phase trajectories connecting them. More complicated stance trajectories can be synthesized by setting the flight time t_f to zero to string multiple actions together.

Efficiently computed kinematic constraint approximations are applied over the entire state tra-

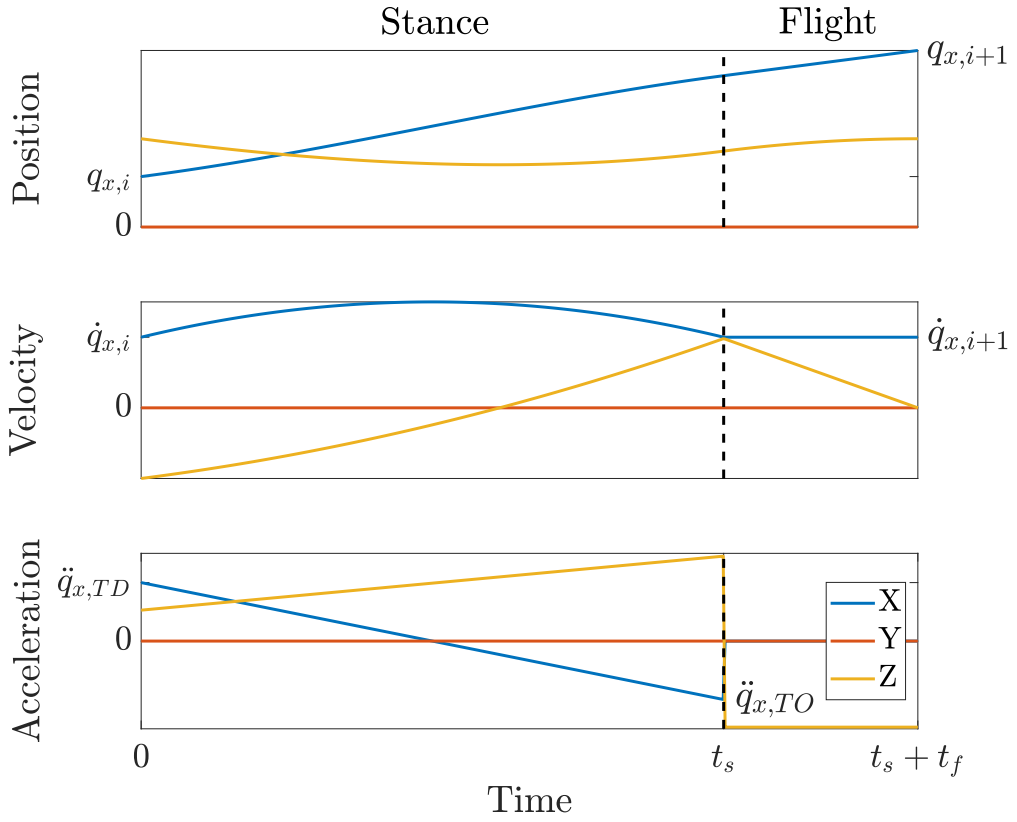


Figure 5: Example state trajectory over one stance and one flight phase. Each planning state is defined as the COM position, orientation, and velocity at the beginning of the stance phase. Each action is defined as a piece-wise linear acceleration trajectory during stance combined with the durations of stance and flight phases, yielding piece-wise cubic and quadratic polynomials for the position and velocity. During the flight phase the robot undergoes ballistic motion until it reaches the beginning of the next stance phase.

jectory. The ground clearance of the corners and center of the underside of the body are checked against a minimum height threshold to avoid collision with the ground. During the stance phase a maximum height threshold of the base of the leg linkage is also applied to ensure the ground is reachable for each leg. No such maximum is applied in flight, allowing the robot to reach terrain of different elevation by leaping up or falling down. These constraints can be written as,

$$h_{body}(\mathbf{q}(t)) \geq h_{min} \qquad \forall t \in [0, t_s + t_f], \tag{2}$$

$$h_{leg}(\mathbf{q}(t)) \leq h_{max} \qquad \forall t \in [0, t_s], \tag{3}$$

where $h_{body}(\mathbf{q}(t))$ and $h_{leg}(\mathbf{q}(t))$ are the clearance of the corners of the underside of the body and

the base of the leg linkages, respectively, and h_{min} and h_{max} are the clearance thresholds.

2.3.2 Action Parameterization

These discrete time states are connected by actions that account for double integrator dynamics and synthesize piece-wise polynomial COM trajectories. Similar polynomial state trajectories have been employed in prior motion planners [Fernbach et al., 2017; Hauser and Ng-Thow-Hing, 2010; Kunz and Stilman, 2014] but have not explicitly handled the intermittent contact of legged locomotion including flight phase dynamics. By specifying piece-wise linear ground reaction forces during stance, the resulting velocity and position trajectories are quadratic and cubic, respectively. Polynomials of this order are useful because cubic splines can efficiently connect states while maintaining dynamic feasibility. Each action is defined as,

$$\begin{aligned}\mathbf{a} &= \begin{bmatrix} \ddot{\mathbf{q}}_{TD}^T & \ddot{\mathbf{q}}_{TO}^T & t_s & t_f \end{bmatrix}^T \\ &= [\ \ddot{q}_{x,TD} \quad \ddot{q}_{y,TD} \quad \ddot{q}_{z,TD} \quad \ddot{q}_{p,TD} \quad \cdots \\ &\qquad \ddot{q}_{x,TO} \quad \ddot{q}_{y,TO} \quad \ddot{q}_{z,TO} \quad \ddot{q}_{p,TO} \quad t_s \quad t_f\ ]^T \end{aligned} \tag{4}$$

where $\mathbf{a} \in \mathbb{R}^{10}$ is the action taken, $\ddot{\mathbf{q}}$ denotes the acceleration of the system, t_s and t_f are the stance and flight time, and the subscripts TD and TO denote touchdown and take-off (start and end of stance times). Under this parameterization, the acceleration, velocity, and position of the system during stance can be written as,

$$\ddot{\mathbf{q}}(t) = (\ddot{\mathbf{q}}_{TO} - \ddot{\mathbf{q}}_{TD})\frac{t}{t_s} + \ddot{\mathbf{q}}_{TD}, \tag{5}$$

$$\dot{\mathbf{q}}(t) = (\ddot{\mathbf{q}}_{TO} - \ddot{\mathbf{q}}_{TD})\frac{t^2}{2t_s} + \ddot{\mathbf{q}}_{TD}t + \dot{\mathbf{q}}_{TD}, \tag{6}$$

$$\mathbf{q}(t) = (\ddot{\mathbf{q}}_{TO} - \ddot{\mathbf{q}}_{TD})\frac{t^3}{6t_s} + \ddot{\mathbf{q}}_{TD}\frac{t^2}{2} + \dot{\mathbf{q}}_{TD}t + \mathbf{q}_{TD}, \tag{7}$$

where $t \in [0, t_s]$ is the time since the beginning of the stance phase. Examples of these trajectories can be seen in Fig. 5. During flight the only acceleration is due to gravity, yielding the following

trajectories,

$$\ddot{\mathbf{q}}(t) = \mathbf{g}, \tag{8}$$

$$\dot{\mathbf{q}}(t) = \mathbf{g}\,(t - t_s) + \dot{\mathbf{q}}_{TO}, \tag{9}$$

$$\mathbf{q}(t) = \frac{\mathbf{g}\,(t - t_s)^2}{2} + \dot{\mathbf{q}}_{TO}\,(t - t_s) + \mathbf{q}_{TO}, \tag{10}$$

where $t \in [t_s, t_s + t_f]$ and $\mathbf{g}$ is the gravity vector. These polynomials can also be reversed in time to enable planning tree growth from the goal towards the start.

Dynamic constraints are easily applied by computing bounds on the valid actions. Rather than bounding accelerations directly, or solving an LP to find acceleration boundaries as in [Fernbach et al., 2017], valid ground reaction forces at the beginning and end of stance are sampled directly, and then transformed into valid accelerations. This transformation is given by Newton's second law,

$$\begin{bmatrix} mI_3 & \mathbf{0} \\ \mathbf{0} & J_p \end{bmatrix} \ddot{\mathbf{q}} = \begin{bmatrix} \mathbf{f}_{GRF} + m\mathbf{g} \\ \tau_p \end{bmatrix}, \tag{11}$$

where $\mathbf{f}_{GRF}$ is the GRF, τ_p is the torque applied about the pitch axis, m and J_p are the mass and pitch inertia of the system, and I_3 is a 3×3 identity matrix. This transformation allows for actuation limits, non-penetration, and friction cone constraints to be respectively enforced by,

$$|\mathbf{f}_{GRF}| \leq f_{max} \tag{12}$$

$$|\tau_p| \leq \tau_{max} \tag{13}$$

$$f_z \geq 0 \tag{14}$$

$$\mu f_z \geq \sqrt{f_x^2 + f_y^2} \tag{15}$$

where f_{max} is a maximum ground reaction force, τ_{max} is a maximum torque threshold, and μ is the friction coefficient. Randomly sampling GRF vectors and torques at touchdown and takeoff within

the bounds of these constraints and applying (74) allows for the automatic generation of feasible actions. The resulting piece-wise cubic position trajectories are then checked at regular intervals for kinematic feasibility to ensure collision avoidance and reachability. Upper and lower bounds are placed on the stance and flight times to aid the planner in producing feasible state-action pairs.

2.3.3 Planning Framework

These state-action pairs form the basis of the tree structure that RRT-Connect employs to explore the state space. While we refer the reader to prior literature for the detailed workings of RRT-Connect [Brandt, 2006; Klemm et al., 2015; Kuffner and LaValle, 2000], the general strategy is to grow trees from the start and goal by alternately *extending* the trees towards randomly sampled states, then attempting to *connect* the trees together. The planning framework proposed here follows the RRT-Connect algorithm from [Kuffner and LaValle, 2000], with particular instantiations of the extend and connect functions to account for the dynamics and kinematics of this problem.

The extend function in any RRT planner guides the planning tree towards new, unexplored areas of the state space by generating a random state, identifying the closest state in the tree, and leveraging a local planner to extend from that closest state towards the random one. The local planner implemented here samples desired states (1) uniformly but with zero angular velocity to regulate orientation. As with KD-RRT [LaValle and Kuffner Jr, 2001], random actions described by (4) are sampled, checking for kinematic feasibility, and returning the closest new state to the desired under a weighted Euclidean distance metric. This extend function is effective at exploring challenging terrain by allowing frequent flight phases.

The connect function handles large portions of less challenging terrain more rapidly than the extend function. This is achieved by computing in closed form the unique cubic spline that connects two states assuming zero flight time and a choosing a stance time that yields a nominal

forward speed. Solving this boundary problem yields the following accelerations:

$$\ddot{\mathbf{q}}_{TD} = -\frac{6\left(\mathbf{q}_{TD} - \mathbf{q}_{TO}\right) - 2t_s(2\dot{\mathbf{q}}_{TD} + \dot{\mathbf{q}}_{TO})}{t_s^2}$$
$$\ddot{\mathbf{q}}_{TO} = \frac{6\left(\mathbf{q}_{TD} - \mathbf{q}_{TO}\right) - 2t_s(\dot{\mathbf{q}}_{TD} + 2\dot{\mathbf{q}}_{TO})}{t_s^2}. \tag{16}$$

The resulting forces are checked with (12–15) and discarded if dynamically infeasible, then checked for kinematic feasibility. If these constraints are satisfied the states are connected. If they are not satisfied, the algorithm considers the feasible portion of the trajectory and inserts a state into the tree corresponding to the midpoint of the feasible portion (to ensure the tree is not trapped right against a constraint). No upper bound is placed on the distance of this connect operation to allow it to traverse large areas of terrain. The long "stance" phase produced by this operation can be thought of not as one single stance phase but as a trajectory induced by a collection of steps taken by the robot between flight phases, during which the lower level controller can easily find a valid whole-body motion.

Another benefit to such a connect function is the ability to exactly connect two states, which is helpful for reducing path length. We employ this ability in two separate ways and compare performance. One common way to reduce path length is to employ RRT*, which adds and removes connections through new states to reduce a cost function [Karaman and Frazzoli, 2010; Klemm et al., 2015]. We implement the re-wiring algorithm described in [Karaman and Frazzoli, 2011], using the connect function to check for valid connections.

Though this method consistently reduces path length, re-wiring the connections whenever a new state is added slows the planner. A simpler short-cutting algorithm offers a faster but sub-optimal method of reducing path length after the completion of the planner [Kuffner and LaValle, 2000]. This algorithm checks each state in the path for connections to other states. If a valid connection is found, those states are directly connected and all intermediate states are removed from the path. This process continues until the goal state is reached. Removing extraneous states in this fashion results in simpler, smoother paths without re-wiring throughout the entire planning

process.

Path quality for this method is further improved by applying an anytime framework, wherein RRT-Connect is called repeatedly and only higher quality paths are accepted [Ferguson and Stentz, 2006]. This anytime framework also improves the speed of the algorithm by restarting the planner if a solution is not quickly found, thus avoiding the increased complexity of adding more nodes to the planning trees.

2.4 Algorithm Analysis

To analyze the proposed planner, the kinematic and dynamic model approximations are validated with a full-order model trajectory optimization framework. We then generate robot paths over several terrains, analyze the resulting performance, and compare to other multi-contact kinodynamic planning algorithms.

2.4.1 Trajectory Validation

Much of the efficiency of the proposed algorithm stems from approximations made to the kinematics and dynamics of legged robot locomotion. These approximations must be sufficiently expressive to capture the dynamic behaviors of interest yet conservative enough for a short-horizon planner to resolve. To test this formulation of the kinematic and dynamic constraints we randomly generate one hundred state-action pairs and check them for feasibility with the extend function described above, employing kinematic and dynamic bounds derived from an MIT Cheetah 3 quadruped model [Bledt et al., 2018]. The resulting trajectories are then passed to a whole-body hybrid trajectory optimization created in the FROST framework [Hereid and Ames, 2017b]. This optimization produces motions that track the planned trajectories over flat ground while satisfying full-order dynamics and kinematics constraints, as well as a DC motor model and friction cones. To avoid solving the contact-implicit problem [Posa et al., 2014], which is quite slow, we specify the contact sequence as all four feet starting on the ground, lifting off simultaneously, then ending within reach of the ground. This is slightly conservative as an ideal footstep planner would select

the footstep sequence best suited to a particular motion, but such a planner is outside the scope of this work.

The optimization framework was able to find solutions for 98% of the tested state-action pairs, indicating that this planner's heuristics closely approximate the full model. Those that it could not find a solution for nearly converged but could not overcome the conservative kinematic restrictions of requiring contact at all four feet during stance, and in each case it is likely that a full local footstep planner would still succeed.

2.4.2 Algorithm Performance

To test the performance of the proposed planner, we created several environments that pose varying motion planning challenges. For each environment we specify a start and goal pose, provide the planner with a height map of the terrain, execute the anytime planner with short-cutting 100 times, and collect statistics. The planner finds a feasible path then executes the short-cutting algorithm but does not re-plan to prioritize algorithm speed. All trials are run in C++ on an Intel Core i7-8700K CPU at 3.7 GHz.

The "Rough Terrain" environment shown in Fig. 6 tests the planner's ability to overcome unstructured terrain including uneven ground, obstacles, and a ledge requiring a flight phase. The "Hallway" environment shown in Fig. 7 tests the ability to generate long-horizon paths in the presence of local minima. The "Slope" terrain shown in Fig. 8 tests the ability to handle steep slopes. Finally, the "Staircase" environment shown in Fig. 4 and 9 puts all these together, requiring a long-horizon, highly dynamic path that navigates slopes and local minima. The Staircase environment was tested with and without an obstacle blocking the middle of the small stairs, requiring the robot to navigate the larger stairs.

Table 1 shows the statistics averaged over each set of trials on the environments including the amount of time spent planning until a feasible path was found, the number of states generated during the planning call, and the length of the returned trajectory. Each environment figure also shows an example trajectory found on that terrain.

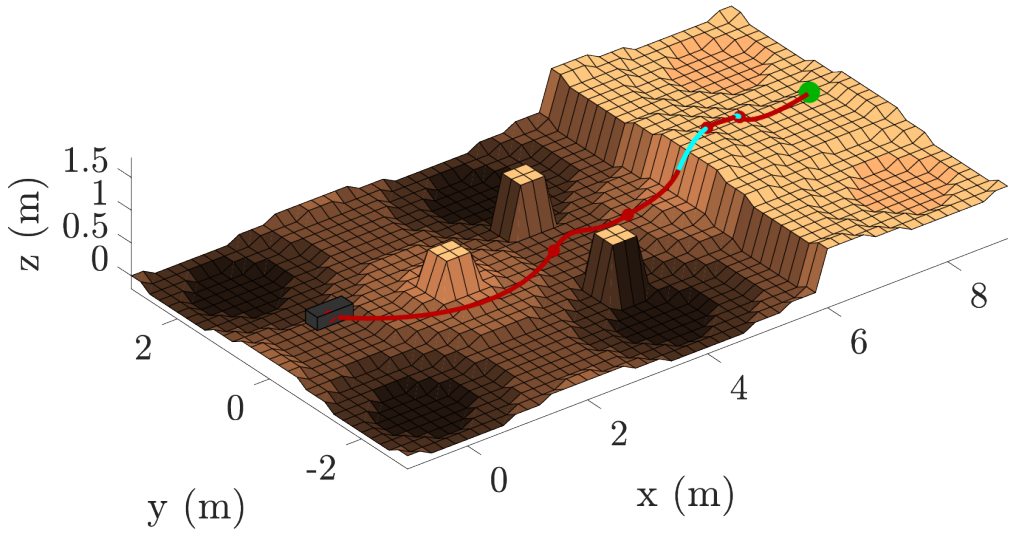


Figure 6: The Rough Terrain environment includes a ledge requiring a flight phase, in addition to uneven terrain and obstacles. In this and the following figures, red lines indicate a stance phase, cyan indicates flight, the black box indicates the starting pose, and the green circle indicates the goal.

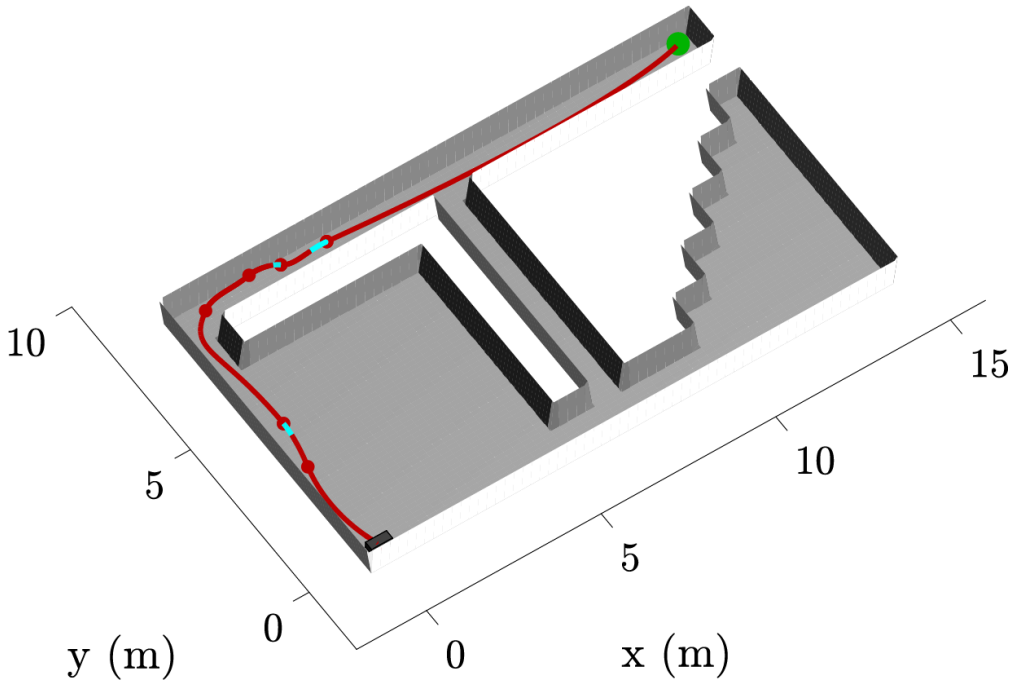


Figure 7: The Hallway environment replicates an indoor areas with long stretches of flat terrain. The local minima presented by the large open portion or the area that approaches but does not reach the goal require a planner with a long enough horizon to avoid.

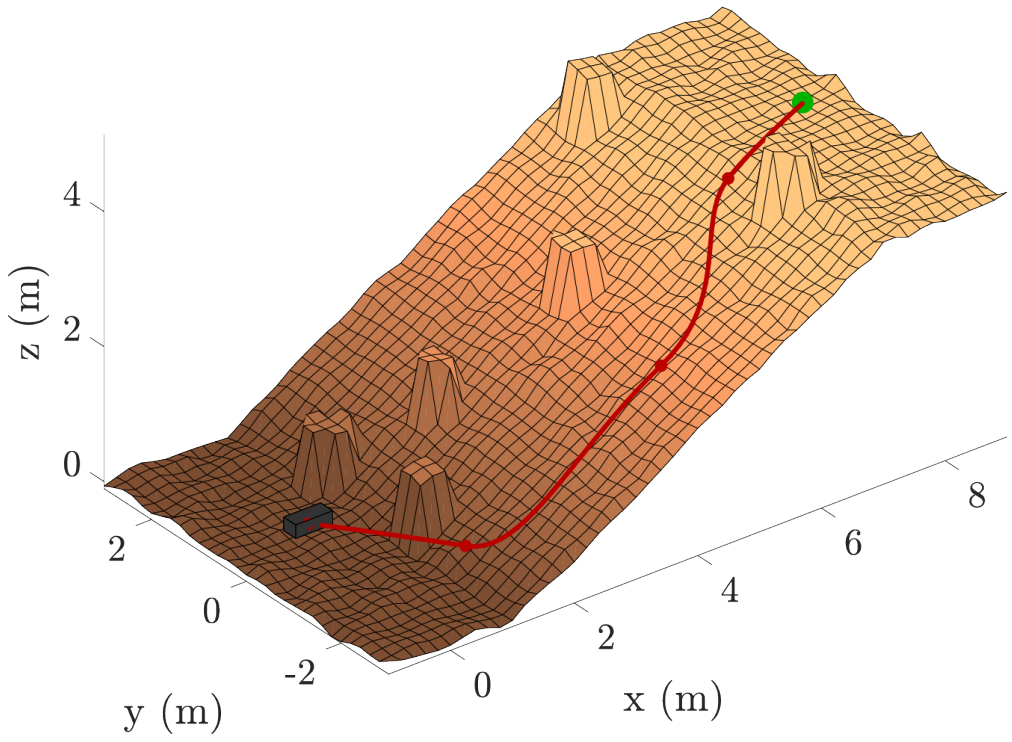


Figure 8: The Slope environment tests the planner's ability to incorporate friction cones and usage of orientation to find more feasible regions of the state space.

The results shown in Table 1 demonstrate the effectiveness of the proposed planner in handling a wide variety of challenging legged robot planning tasks, as every planning call was successfully completed. The length of each plan is on the order of tens of body lengths, showcasing a horizon long enough for global planning. Each plan – with the exception of the Staircase with the obstacle – is computed on the order of seconds and therefore fast enough for real-time deployment. The Hallway environment in Fig. 7 showcases the connect function's ability to rapidly traverse wide sections of trivial terrain and return a straightforward path with little erratic motion while avoiding local minima. Conversely, the performance on the Rough Terrain environment demonstrates the ability to navigate dynamically challenging terrain with flight phases which are unresolvable by geometric planners.

These challenges are combined in the Staircase environment. The algorithm automatically discovers that the regular stairs are more traversable than the large stairs and explores that area rapidly, with no scripted stair-climbing controller. This behavior is desirable as favoring less challenging

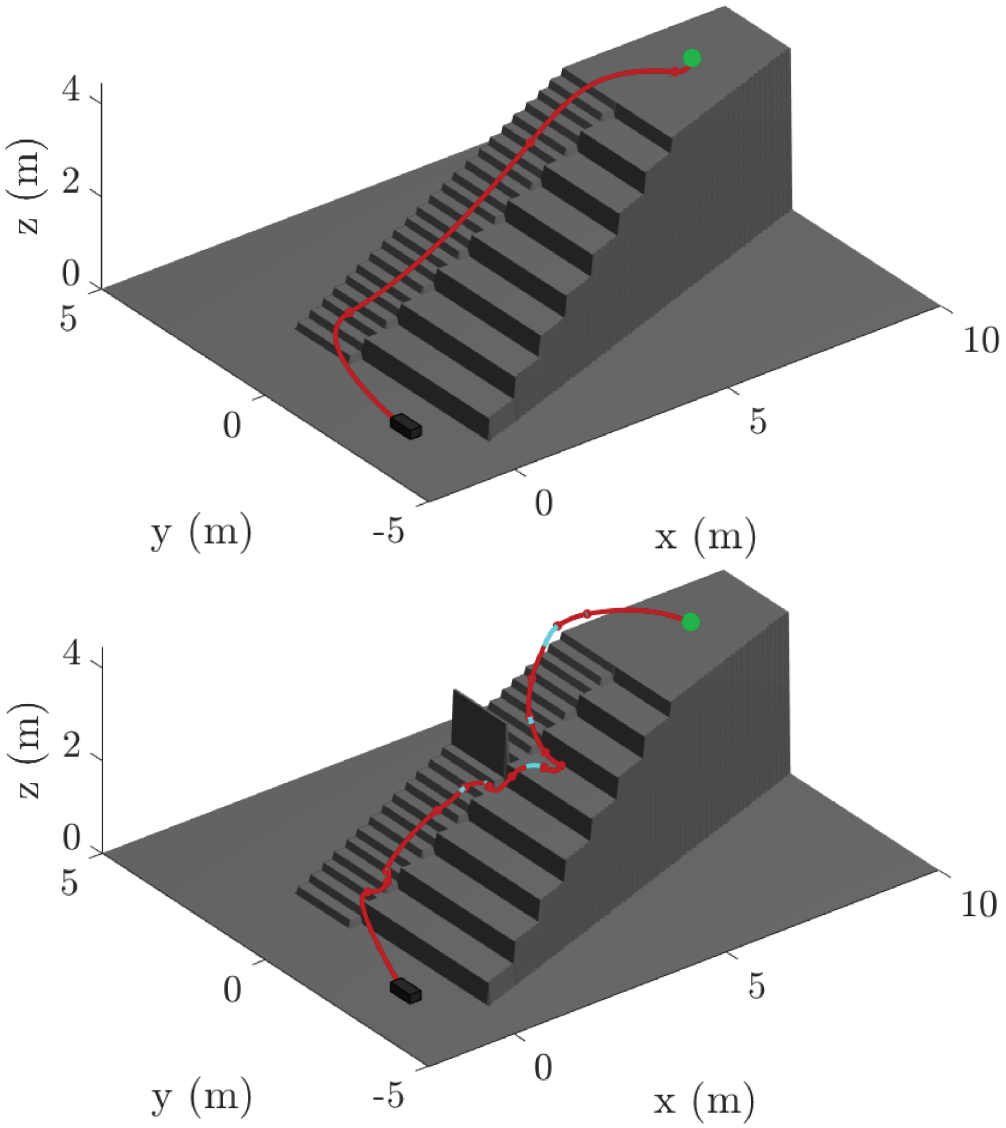


Figure 9: The Staircase environment, top, based on the location from Fig. 4, presents a long-horizon task with discrete changes in ground height. When an obstacle is placed on the regular stairs, bottom, the planner must find a dynamic path over the larger steps.

motion improves the likelihood of resolution by a lower level controller. When those stairs are obstructed, the algorithm employs multiple flight phases to leap up the large stairs towards the goal, although it takes much longer to find such a path through the tightly constrained space.

The two methods of reducing path length – RRT*-Connect and anytime RRT-Connect with short-cutting – are tested on the Slope terrain environment in Fig. 8 and allowed to run for five seconds before termination. Each planner is called one hundred times and the results are averaged.

Table 1: Algorithm performance on test environments. Data reported as mean $\pm$ standard deviation over 100 trials.

Environment	Plan Time (s)	States Generated	Plan Length (m)
Rough Terrain	2.3 ± 2.2	1100 ± 1000	14 ± 2.8
Hallway	1.8 ± 1.4	640 ± 420	25 ± 2.8
Slope	0.15 ± 0.2	150 ± 31	13 ± 2.1
Staircase	2.9 ± 2.5	720 ± 550	17 ± 2.6
Staircase w/obstacle	26 ± 20	4400 ± 2700	21 ± 3.8

The results are shown in Fig. 10, along with the average path length for regular RRT-Connect with short-cutting.

Figure 10 demonstrates a small advantage of anytime RRT-Connect with short cutting over RRT*-Connect in quickly reducing path length, though both perform well compared to the non-optimal planners. The anytime framework is able to reduce cost by rapidly growing multiple trees in succession rather than re-wiring a single tree. Both of these methods outperform standard RRT-Connect in less than one second, demonstrating the utility of these cost-reducing strategies in yielding high quality paths.

2.4.3 Algorithm Benchmarking

We compare the proposed algorithm directly against other state-of-the-art methods on the "Plinth" environment, based on [Geisert et al., 2019], shown in Fig. 11. In particular we benchmark against the root trajectory from the multi-contact RRT-Connect planner described in [Fernbach et al., 2017] and implemented in [Geisert et al., 2019] as well as the contact-implicit trajectory optimization method presented in [Winkler et al., 2018a]. Each test is performed with a model of the ANYmal quadruped [Hutter et al., 2016]. The proposed algorithm and the trajectory optimization are executed on the CPU described above, and the statistic for the multi-contact RRT-Connect planner is obtained directly from [Geisert et al., 2019] which utilized a similar CPU.

The results from the benchmarking are shown in Table 2. The proposed algorithm constructs plans six times faster than the multi-contact RRT-Connect method when similarly constrained to

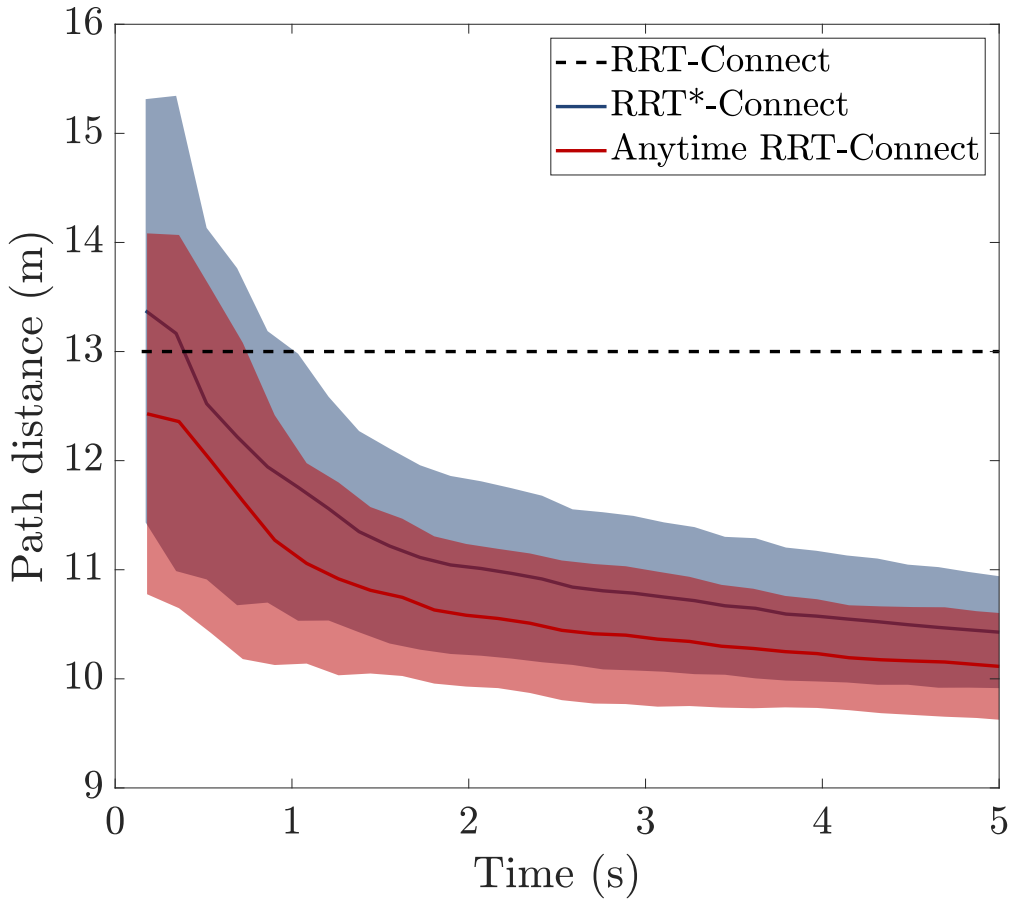


Figure 10: Path length reduction is tested on the Slope environment for three variations of the proposed planner. Both RRT*-Connect and anytime RRT-Connect provide higher quality paths over time than regular RRT-Connect, with the anytime version outperforming its RRT* counterpart on account of its speed. RRT-Connect returns after 0.15 seconds on average but does not further improve path quality. The solid lines indicate the mean and the shaded regions indicate one standard deviation over 100 trials.

planar motion, and is still slightly faster when allowed to explore the environment in three dimensions. This algorithm is also around an order of magnitude faster than the trajectory optimization method, although that method computes footstep locations in addition to a pose trajectory. This suggests that such a method could be an effective short-horizon planner to determine contact locations a few steps ahead once provided the global plan from the algorithm presented here.

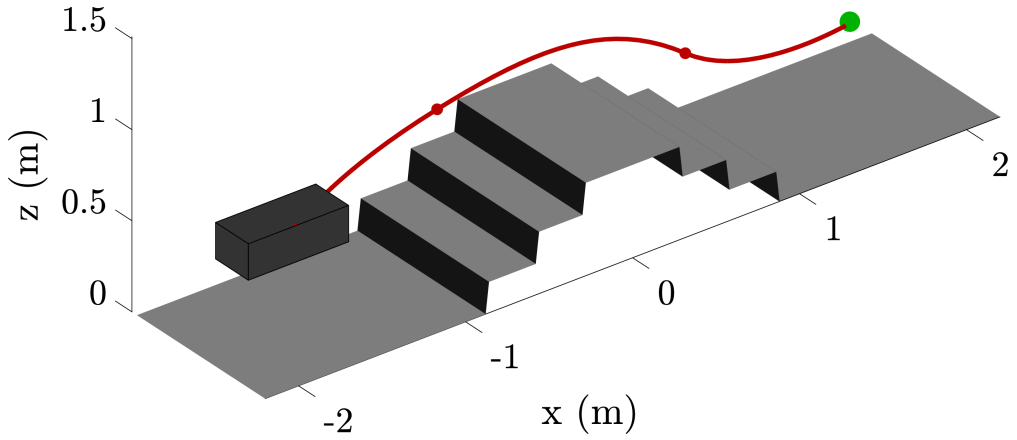


Figure 11: The algorithm benchmarking is performed on the Plinth environment, which requires navigating height changes but does not explicitly require a flight phase.

Table 2: Algorithm comparison on Plinth environment. Data reported as mean $\pm$ standard deviation over 100 trials.

Solver	Dimension	Plan Time (s)
Proposed Algorithm	2D	0.20 ± 0.21
Proposed Algorithm	3D	1.0 ± 0.91
Prior RRT-Connect [Geisert et al., 2019]	2D	$1.3 \pm$ unreported
Traj Opt (w/contacts) [Winkler et al., 2018a]	3D	9.5 ± 0.082

2.5 Discussion and Conclusion

The diversity of the environments tested showcases the speed and ability to dynamically overcome obstacles of the proposed algorithm. The trajectory validation results and optimality analysis indicate that the resulting paths are feasible and near-optimal. These features together highlight the practicality of this algorithm for deployment as a global planner for legged robots.

By design, many of the assumptions and model approximations made in this planner are heuristic in nature and thus offer no model-based guarantees. The strength of this approach is rooted in the objective of the planner – finding a basic path that navigates an unstructured environment which can serve as an initial seed for short horizon footstep and whole-body planners. Relaxing hard constraints on full-order feasibility allows for each constraint to be expressed in a more computationally efficient way. This enables longer planning horizons, ensuring that the footstep and whole-body planners can focus on refining motion rather than trying to escape local minima. In

particular, reasoning about contact via the net ground reaction force rather than through each individual contact captures the hybrid nature of intermittent contact but does not enforce particular contact locations. This approach allows a shorter horizon footstep planner with a more expressive contact model to dedicate increased computation to selecting robust contact locations.

Another consideration of this approach is the role of a discrete number of motion primitives which define the action space employed by the planner. A potential drawback of implementing a discrete union of continuous action spaces is how to handle the combinatorial nature of selecting these primitives, particularly for larger numbers of potential primitives. This is a known issue in fields such as manipulation where behavior libraries must be quite large to capture the desired object manipulations. However, the primitives chosen here are not attempting to discretely capture the space of all possible legged robot behaviors, but rather a hierarchy of model fidelity approaching a full expression of the platforms kinematics and dynamics. One could presumably add more primitives which captured further details of the system and its interactions with the environment, however the two presented here are sufficient to expand the capabilities of the platform as evidenced by the performance in a diverse range of environments.

While the centroidal dynamics approximations of this model are reasonably accurate for low leg inertia quadrupeds, these approximations may be invalid for systems with high-inertia limbs. This framework also does not explicitly reason about body stability and instead assumes that the ground reaction force vector always extends through the COM. For bipedal robots or other highly underactuated systems this assumption may not hold. Accounting for stability would require either a method of counteracting the moment caused by the GRF, or by expanding this planning framework to explicitly reason about the stability of the body.

3 Quad-SDK: Open-Sourcing Autonomous Agility via a Full Stack Software Framework

Section 2 introduced a novel global motion planning algorithm along with results demonstrating its efficiency in a number of unstructured environments, but attaining agile autonomy in these environments requires more than just a good approximation of the desired motion. Executing these plans on legged platforms requires systems that can reliably translate them into actuator inputs while parsing sensor streams, handling full-order dynamics and constraints, and resolving any disturbances. However, prior examples of such frameworks generally rely on some offline or manual trajectory generation, and are often only implemented for one system which reduces the utility for related locomotion research. This section describes Quad-SDK[1], a set of open-source software packages created to provide this functionality in a unified hierarchical framework for multiple quadrupedal platforms and available as a resource for the broader legged locomotion community [Norby et al., 2022b].

3.1 Introduction

3.1.1 Prior Work Towards Agile Autonomy

As discussed in Section 1, unlocking the potential of legged robot locomotion requires improving the agility and autonomy of a system. Over the past few decades many efforts have extended the agility of legged robots. In the 1980s Marc Raibert introduced a robot design with spring-like legs and simple control laws that decoupled motion to maintain dynamic stability. These breakthrough ideas enabled remarkably dynamic motions like running, jumping, and even acrobatics as shown in Figure 12 [Hodgins and Raibert, 1988; Raibert, 1986]. The Raibert hopper and controller inspired many subsequent developments in robot agility due to their simple efficacy. Work with robots like

[1]Quad-SDK is a collaborative project built by a hardworking team of researchers at CMU and as such I cannot take complete responsibility for its development. My contributions to the project are as follows: project management, development of Global Planner, logging, visualization, and communications packages, co-development of Local Planner, Robot Driver, and simulation packages.

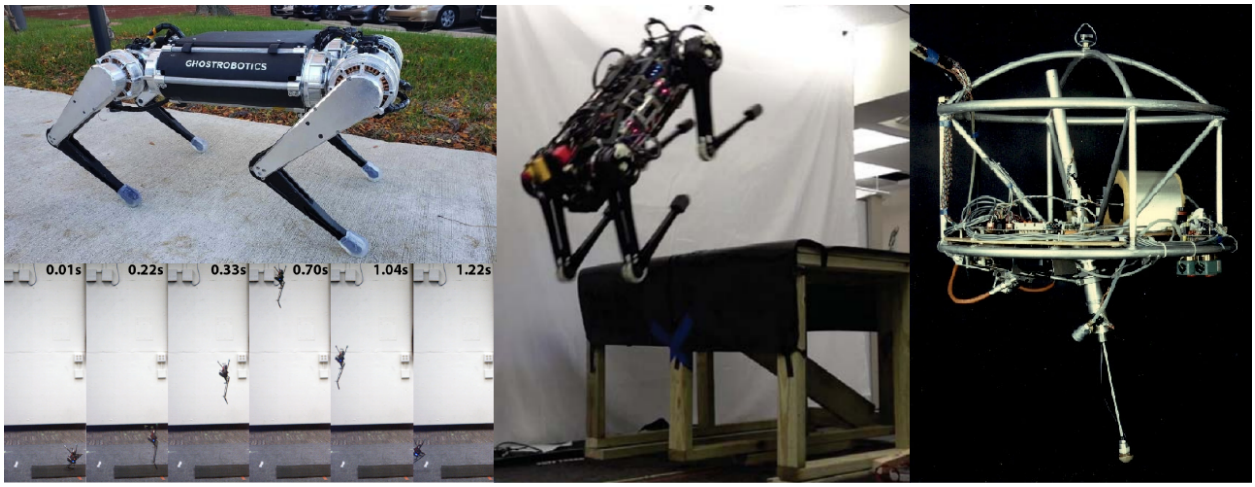


Figure 12: Robots such as the Ghost Robotics Spirit 40 (top left, from [Ghost Robotics, 2020]), SALTO 1-P (bottom left, reproduced from [Haldane et al., 2017]), the MIT Cheetah 3 (center, reproduced from [Nguyen et al., 2019]), and the Raibert Hopper (right, reproduced from [Hodgins and Raibert, 1988]) demonstrate a remarkable capacity for agility despite the difficulty they face in completing fully autonomous navigation tasks in unstructured environments.

RHex, SCOUT II, and DASH expanded this agility to rough terrain by embracing these spring-like legs and their accompanying spring-loaded inverted pendulum (SLIP) model-based control [Birkmeyer et al., 2009; Blickhan, 1989; Poulakakis et al., 2006; Saranli et al., 2001], or by controlling the legs to behave in a SLIP-like way as with MABEL and ATRIAS [Hubicki et al., 2016; Sreenath et al., 2012]. SLIP-based robots demonstrate remarkable stability even during relatively agile motions, but since the dynamics of the model itself are fairly restrictive they struggle to perform more complicated motions. For example, RHex is capable of performing acrobatic behaviors, but doing so requires an entirely new modeling framework [Johnson and Koditschek, 2013b]. Other robots that employ Raibert-like controllers show impressive agility such as Big Dog or SALTO without SLIP-based modeling [Haldane et al., 2017; Playter et al., 2006] yet these platforms still struggle to navigate with full autonomy because their underlying controllers do not intelligently account for their environment in prescribing a desired motion.

This highlights the fundamental issue with simplistic approaches to dynamic locomotion control. These models often make the implicit assumption that locomotion is periodic and that any deviations from the desired limit cycle (e.g. changes in ground height) are disturbances that ought

to be rejected once they occur. Yet often navigating rough terrain with agility requires planning ahead to overcome these deviations (e.g. accelerating a few steps before a change in ground height to prepare for a leap). Even if these reactive frameworks were coaxed into achieving desired behaviors by modifying their nominal limit cycles as in [Bazeille et al., 2014], they offer no clear method for autonomously doing so for dynamic motions over arbitrary terrains. In lieu of these methods, autonomous agility in unstructured environments can be promoted with some form of motion planning.

Motion planning for legged robots in rough terrain is a well-studied field – the kinematic methods and hierarchical control structure Robert McGhee et al. demonstrated on a hexapod robot predate Raibert's work [McGhee and Iswandhi, 1979; Orin et al., 1976]. Since McGhee, legged robot motion planning has developed to include more complicated models, planning methods, and terrain. The DARPA Learning Locomotion project in 2011 pushed these methods to new levels, with groups implementing a variety of methods from trajectory optimization to search based planning to machine learning for quadrupedal locomotion over extremely rough terrain [Byl et al., 2009; Kalakrishnan et al., 2011; Zucker et al., 2011]. Each of these teams – like McGhee – adopted some sort of hierarchical planning and control structure to separate high-level planning from low-level execution, thereby enabling real-time operation.

However in order to achieve real-time speeds and increase error margins of low-level controllers, many of these and more recent high-level planners restrict the dynamics by assuming quasi-static or Zero-Moment Point (ZMP) stability [Carpentier and Mansard, 2018; Fankhauser et al., 2018; Hornung et al., 2012; Tonneau et al., 2018; Vukobratović and Borovac, 2004]. These restrictions prohibit flight phases or other highly dynamic motions, greatly reducing the agility of the robot platform. Other recent methods of motion planning have relaxed these restrictions by planning with models of the full dynamics or just the centroidal dynamics of a robot, but these methods remain too computationally expensive for real-time execution over long time horizons [Dai et al., 2014; Mombaur, 2009; Posa et al., 2014]. Other researchers have leveraged further simplifications of either the kinematics and dynamics to achieve more agile autonomy in real-time

[Di Carlo et al., 2018; Neunert et al., 2018; Winkler et al., 2018a], but in doing so are still restricted to planning over very short horizons (generally under one second) or over terrain with a known structure.

A few more recent works have demonstrated some ability to plan and execute dynamic primitives such as leaps [Chignoli and Kim, 2021; Gilroy et al., 2021; Nguyen et al., 2019; Park et al., 2015]. However, these works either rely on assumptions about the structure of the environment, offline generation of full-order trajectories for tracking (and therefore coming to rest before leaping), or manually authored reference trajectories for online optimization. The lack of full integration with an agility-enabling global planner and a low-level controller capable of executing these behaviors prohibits truly autonomous agility. Additionally, the success of these frameworks can often be dependent on the details of the implementation of the hierarchy, which so far has only developed for isolated demonstrations rather than as tools for use by the wider legged locomotion community.

3.1.2 Prior Work in Software Tools for Legged Robots

The agile autonomy these works approach would be ideal for expanding performance in real world applications like environmental monitoring, industrial inspection, disaster recovery, and material handling [Bellicoso et al., 2018; Hutter et al., 2017; Kolvenbach et al., 2020]. Developing robotic solutions for these applications is more effective when researchers can focus on higher level behaviours instead of low-level implementation. Infrastructure to enable this sort of development is well-established for simpler platforms, e.g. MoveIt for manipulators [Coleman et al., 2014] or ROS Navigation for planar mobile robots [Marder-Eppstein et al., 2010], but they are not suited to handle the complexity of legged robots.

In the absence of standard tools for a full quadruped autonomy and control stack, many open-source software tools have been created to solve various individual layers of the locomotion problem. For example, tools like Altro [Howell et al., 2019], Crocoddyl [Mastalli et al., 2020], OSC2 [Farshidian et al., 2022], and TOWR [Winkler et al., 2018b] are primarily focused on implementations of constrained optimal control problems. Other packages like Drake [Tedrake and the Drake

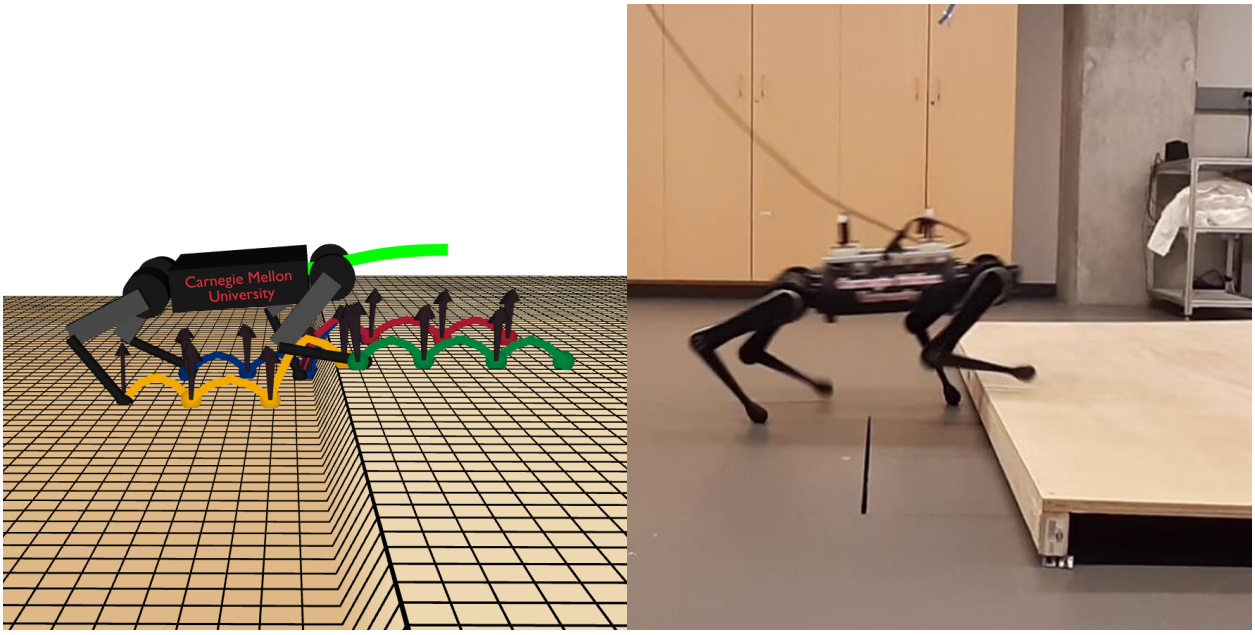


Figure 13: Quad-SDK enables agile autonomy for quadrupedal systems in unstructured terrain. Left: live data visualization provided by Quad-SDK exposes the underlying planning and control states. Right: deployment of Quad-SDK on a hardware platform.

Development Team, 2019] and FROST [Hereid and Ames, 2017b] are targeted towards design, simulation, and optimization for multi-body systems but do not provide full stack support. Robot manufacturers like Unitree, ANYbotics, and Boston Dynamics provide platform-specific SDKs for external development, but these tools are generally closed source and platform specific. Some packages such as Cheetah-Software [DiCarlo et al., 2019], Free Gait [Fankhauser et al., 2019], and CHAMP [Jimeno] offer open-source hierarchical frameworks for legged robots, but they lack the desired combination of high-level autonomy, agility, and support for multiple platforms.

Quad-SDK meets the needs of modern locomotion researchers in two ways. First, it provides autonomous agility for quadrupeds through advanced algorithms including the novel global motion planner described in Section 2 that can plan long-horizon motions including aerial phases, as well as an efficient real-time Nonlinear Model Predictive Controller (NMPC) for executing agile behaviors. Second, it enables rapid development of new locomotion algorithms through its modular architecture, support for multi-robot simulation, and a host of visualization and data-processing tools. Section 3.2 discusses the structure of the framework and the tools it provides, Section 3.3

demonstrates the system in action, and Section 3.4 discusses plans for future extensions of the framework.

3.2 Software Architecture

Quad-SDK provides a modular hierarchical structure to both enable system modification and to match the separation of timescales present within legged locomotion. This structure is illustrated in Figure 14 and is divided into three primary sections – Global Planner, Local Planner, and Robot Driver – each of which are implemented as ROS nodes which wrap a C++ class. This structure enables asynchronous communication to accommodate timescale separation and sensor update rates and allows users to easily extend the software by implementing their own classes inside these nodes or packages that communicate over the same topics. This extensibility is supported with API documentation, continuous integration, and unit testing to enable anyone from researchers to companies to use and contribute to Quad-SDK under an open source license[2]. The following subsections describe the functionality of the primary components in greater detail as well as the development tools which accompany the SDK.

3.2.1 Global Planner

The top of the stack contains the Global Planner, which computes collision-free trajectories of the robot body that guide the system from its current state to the goal state given a map of the terrain [Norby and Johnson, 2020]. Unlike standard ROS Navigation planners, this system explicitly handles dynamics, constraints, and aerial phases for legged systems as well as 2.5D terrain information through existing packages [Fankhauser and Hutter, 2016]. This allows the robot to plan ahead to avoid future failures and enables greater autonomy than direct user-provided twist inputs, although Quad-SDK supports twist commands as well. The wrapper for this algorithm subscribes to the newest desired goal state and terrain data, and generates new plans at 10 Hz to quickly respond to any changes and to converge to more optimal solutions. The current software loads pregenerated

[2]Available at `https://github.com/robomechanics/quad-sdk`.

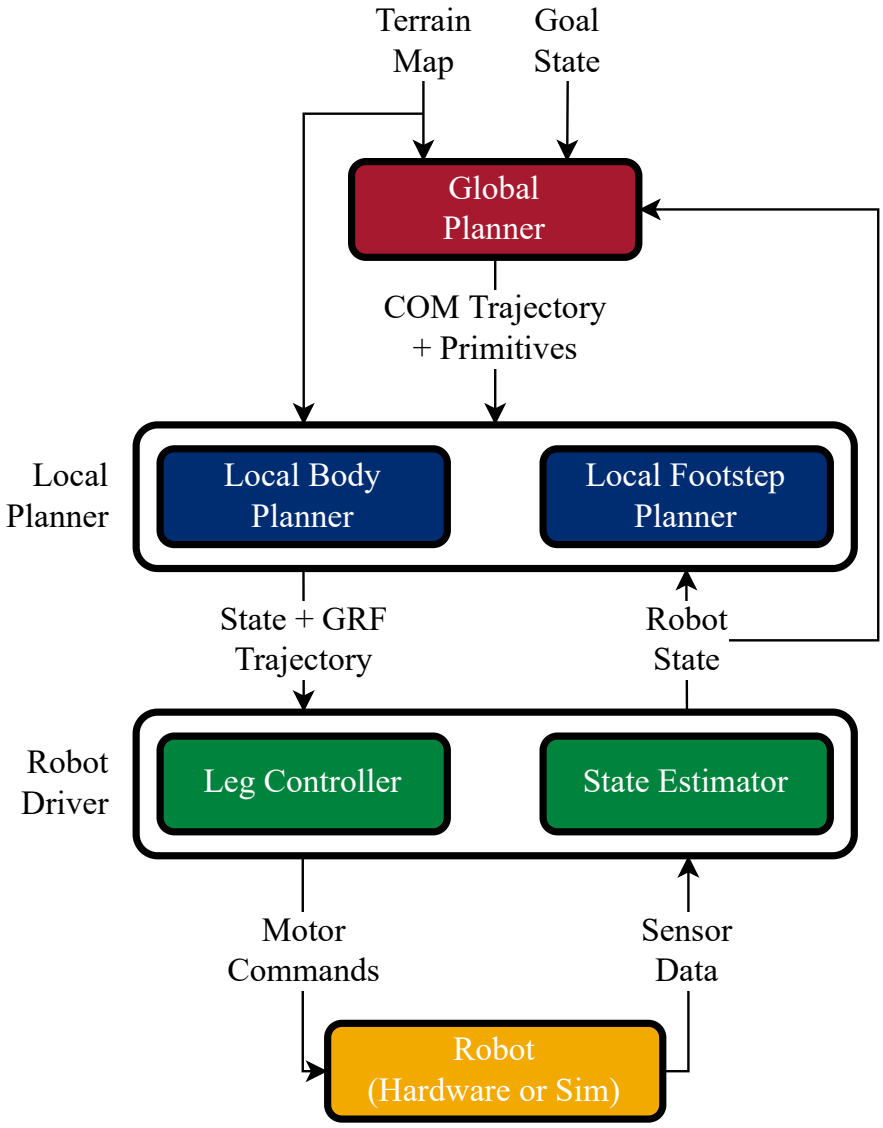


Figure 14: The hierarchical structure of Quad-SDK is determined by the timescales observed in legged locomotion. The Global Planner (red, 10 Hz) reacts to updates in goal state or terrain information, the Local Planner (blue, 100 Hz) closes the loop on the state of the robot body, and Robot Driver (green, 500 Hz) maintains smooth joint-level control and state updates as well as communication with the robot (gold, platform dependent frequency).

terrain data but future releases will support perception packages for deployment in unstructured terrain. The resulting trajectory is then augmented with any motion primitive information (such as trot or leap) before being passed to the Local Planner for tracking.

3.2.2 Local Planner

Given this global plan as a reference trajectory or twist commands from a joystick or keyboard, the Local Planner determines the contact timing, locations, and forces to execute the plan. The Local Planner is divided into two parts: the Local Body Planner which plans the body motion, and the Local Footstep Planner which computes the contact sequence, footstep locations, and swing foot trajectories. While many promising works have combined these two systems [Cleac'h et al., 2021; Winkler et al., 2018b], keeping them separate and iteratively sharing solutions enables specialized algorithms that can solve the problem much faster and increase overall system responsiveness.

The Local Body Planner uses NMPC to determine the ground reaction forces to best track the nominal body trajectory, similar to [Ding et al., 2021]. This NMPC reduces the complete robot to a single rigid body model, but maintains the nonlinearity of SE(3) dynamics. The system input is treated as the ground reaction forces u_i, and the hybrid system is simplified to a switched system with a clock-based contact schedule. Given the desired body state trajectory $x_{i,ref}$, nominal ground reaction force $u_{i,ref}$, planned foot position p_i, and initial condition x_{int}, the solution of discrete-time MPC can be formulated as the following optimization problem:

$$
\begin{aligned}
\min_{x,u} \quad & \sum_{i=0}^{N-1} \|x_{i+1} - x_{i+1,\text{ref}}\|_{Q_i} + \|u_i - u_{i,\text{ref}}\|_{R_i} \\
s.t. \quad & x_0 = x_{\text{int}} \quad \text{(initial condition)} \\
& f\left(x_i, x_{i+1}, u_i, p_i, t_i\right) = 0 \quad \text{(dynamic model)} \\
& x_i \in \mathbb{X} \quad \text{(state bound)} \\
& u_i \in \mathbb{U} \quad \text{(control bound)} \\
& C_i u_i \leqslant 0 \quad \text{(friction pyramid)} \\
& D_i u_i = 0 \quad \text{(contact selection)}
\end{aligned}
\tag{17}
$$

where $i \in [0, \dots N-1]$, Q_i and R_i are the diagonal quadratic cost matrix for state and input, t_i is the finite element duration, function $f(\cdot)$ is the implicit dynamics, $\mathbb{X}$ and $\mathbb{U}$ are the feasible

state and control set, C_i is the friction pyramid matrix, and D_i is the contact selection matrix. The nonlinear program is constructed by the automatic differentiation from CasADi [Andersson et al., In Press, 2018] and solved with IPOPT [Wächter and Biegler, 2006]. Despite this nonlinear formulation, we achieve update rates over 100 Hz for horizons of two or more gait cycles with 16 elements per period through a novel warm starting approach. In particular, we vary the duration of the first finite element to allow subsequent knot points to remain stationary in time. This greatly improves the quality of the warm start and decouples the time scale between the NMPC update rate and the finite element discretization, allowing faster solves over longer horizons.

The Local Footstep Planner uses the most recent local body plan to update predictions of foot trajectories. It first computes a contact schedule given a nominal gait or global plan primitive information. It then selects discrete foothold positions according to this schedule. Similar to Raibert's heuristic [Kim et al., 2019; Raibert, 1986], we use the local plan to determine nominal foothold positions p_{nominal} based on dynamic and kinematic heuristics as

$$p_{\text{nominal}} = p_{\text{center}} + p_{\text{vel}} + p_{\text{centrifugal}} \tag{18}$$

where

$$\begin{aligned} p_{\text{center}} &= \underset{p}{\operatorname{argmin}} \left(\max_{i \in \text{st}} \|p - p_i\|_2^2 \right) \\ p_{\text{vel}} &= \sqrt{\frac{z_0}{\|g\|}} \left(v_{\text{touchdown, ref}} - v_{\text{touchdown}} \right) \\ p_{\text{centrifugal}} &= \frac{z_0}{g} v_{\text{touchdown}} \times \omega_{\text{ref}} \end{aligned} \tag{19}$$

Specifically, a minimum enclosing circle problem is formulated for the leg base positions for each stance phase and solved by the Welzl's algorithm [Welzl, 1991] to compute p_{center}, which ensures foothold reachability. Offset terms p_{vel} and $p_{\text{centrifugal}}$ based on velocity and angular velocity [Kim et al., 2019] tracking are added to the nominal foot position to minimize undesired moments caused by ground reaction forces during agile motion. This nominal foothold is then refined with a local traversability search similar to [Fankhauser et al., 2018] to improve robustness to uncertainty in foot tracking and friction. Finally, these footholds are interpolated with a cubic Hermite spline

based on desired foot retraction and the terrain heights at liftoff, touchdown, and swing apex to obtain swing leg trajectories.

3.2.3 Robot Driver

The Robot Driver is in charge of interfacing with the robot to ensure that these plans are executed as accurately as possible. This is accomplished through a Leg Controller, which parses the local plan to select the correct joint torques to apply, and a State Estimator, which gathers all the relevant sensor streams to construct an estimate of the full state of the robot. Since the timescales of motor and sensor dynamics are quite fast (bandwidths typically over 500 Hz), both systems run in the same thread to reduce latency and improve communication reliability. This system directly interfaces with either the simulator or hardware for straightforward sim-to-real transfer.

Leg Controller converts the trajectories and controls from the local planner for stance and swing legs into joint space commands. The local plan is interpolated and passed through inverse kinematics to generate generalized coordinate reference positions q_{ref} and velocities $\dot{q}_{\mathrm{ref}}$, as well as GRFs u_{ref} for each leg. These are then mapped to feedforward motor commands in joint (generalized) space by inverse dynamics. This is performed by solving for the feedforward stance torques $\tau_{\mathrm{ff,st}}$ and swing torques $\tau_{\mathrm{ff,sw}}$ with

$$\tau_{\mathrm{ff,st}} = -J_{\mathrm{st}}^T u_{\mathrm{ref}} \tag{20}$$

$$M\ddot{q} + h + \begin{bmatrix} 0 & (J_{\mathrm{st}}^T u_{\mathrm{ref}})^T & 0 \end{bmatrix}^T = \begin{bmatrix} 0 & \tau_{\mathrm{ff,st}}^T & \tau_{\mathrm{ff,sw}}^T \end{bmatrix}^T \tag{21}$$

$$J_{\mathrm{st}}\,\ddot{q} + \dot{J}_{\mathrm{st}}\,\dot{q} = 0 \tag{22}$$

where q is the system states in articulated body generalized coordinates, M is the inertia matrix, h is the sum of Coriolis and potential terms, and J_{st} is the block of the kinematic Jacobian matrix for

the stance legs. These terms are combined with joint space feedback to obtain the control law

$$\tau = \tau_{\text{ff}} + \tau_{\text{fb}} \tag{23}$$

$$\tau_{\text{fb}} = K_p \left(q_{\text{ref}} - q \right) + K_d \left(\dot{q}_{\text{ref}} - \dot{q} \right) \tag{24}$$

$$\tau_{\text{ff}} = \begin{bmatrix} \tau_{\text{ff,st}}^T & \tau_{\text{ff,sw}}^T \end{bmatrix}^T \tag{25}$$

where K_p and K_d are proportional and derivative gains.

The State Estimator is responsible for parsing sensor streams and maintaining a high-frequency estimate of the full state of the robot. Currently this class performs sensor fusion of motion capture, IMU, and encoder data to generate this estimate. The body position and orientation are obtained directly from motion capture, while angular velocity and joint information are read from the IMU and motor encoders. Linear body velocity is computed with a complementary filter which fuses the differentiated body position and integrated IMU linear acceleration. This complementary filter high-passes the IMU estimate and low-passes the motion capture estimate at a cutoff frequency of 5 Hz, indicating that the bulk of the estimation required for highly dynamic maneuvers is provided by the onboard IMU. Future releases will support fully onboard algorithms such as an extended Kalman filter (EKF) similar to [Bloesch et al., 2013], which will enable outdoor deployment.

In simulation, a Gazebo plugin provides ground truth state and contact information. The body position, velocity, and orientation as well as joint and foot positions and velocities are then published as a robot state topic.

3.2.4 Development Tools

Quad-SDK comes with a number of tools to enable rapid development of algorithms and applications for legged robots. The Gazebo simulation accompanying Quad-SDK allows users to interface with one or multiple quadrupeds through teleoperation and point-to-point navigation within a diverse set of environments. Online data visualization of the robots, terrain, global and local plans, foot trajectories, and ground reaction forces is provided through RViz [Hershberger et al.]. Users

may interact with the visualization interface to select the goal states for navigation. Real-time display of the data is provided through a Plotjuggler interface [Faconti]. Quad-SDK also comes with a set of tools to collect and post-process data. Logging scripts enable recording of data via rosbags for easy playback and debugging. MATLAB scripts are included to produce publication-ready figures from these logs, as shown in Section 3.3.

3.3 Performance Tests

This section presents the results of several tests of the functionality of Quad-SDK in both simulation and hardware. In each test, the robot was tasked with finding and executing a plan from its current state to the goal state. In all but the robustness test the robot was provided with a ground-truth map of the terrain as well as ground-truth state information to isolate the performance of the planning and control framework. All simulation tests were performed using Gazebo 9 with the ODE physics engine, and all processes were executed on a machine running Ubuntu 18.04 with an Intel Core i7-12700K CPU @ 4.9 GHz and 64 GB of RAM. Hardware tests were performed with a Ghost Robotics Spirit 40 quadruped platform. Global and Local Planner were executed on a remote laptop running Ubuntu 18.04 with a Intel Core i7-10750H CPU @ 2.6 GHz and 16 GB of RAM, while the Robot Driver was executed on the onboard Nvidia Jetson TX2. Communication between the robot and the remote computer was handled via Ethernet to reduce dropout, but otherwise the robot was untethered and running on its own power.

3.3.1 Local Minima and Leaping Environment

This task tests the ability of Quad-SDK to navigate local minima as well as perform an agile leaping behavior. A simulated robot is provided with a terrain map that includes infeasible regions and a 15 cm step which requires a leap. The goal state provided is a few dozen body-lengths away, and in a location such that greedily navigating towards the goal would cause immediate failure.

The Global Planner takes around 50 milliseconds to find a feasible route around these local minima to the goal state (Fig. 15), and constantly replans while executing the motion to find shorter

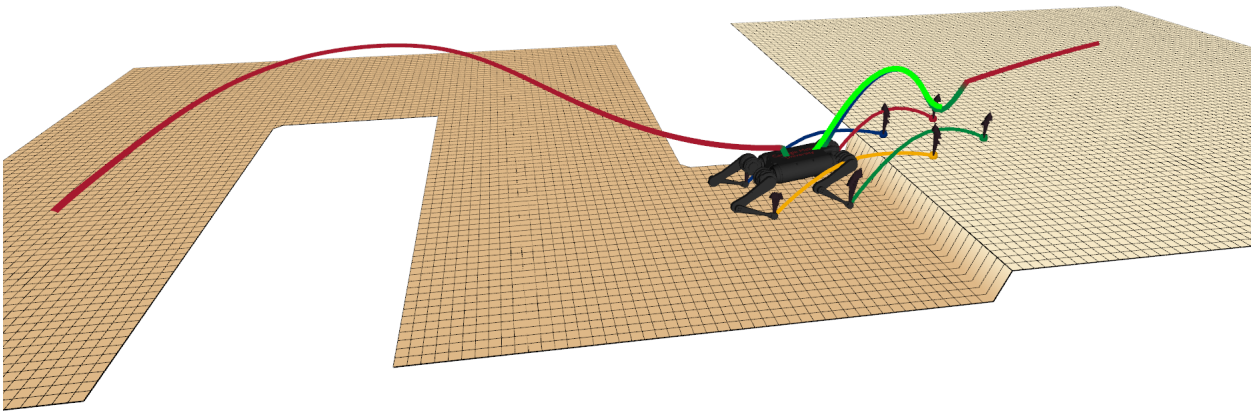


Figure 15: Local Minima and Leap Environment – Global Planner. The Global Planner efficiently solves navigation tasks that require long-horizon plans and agile behaviors such as flight phases. The global plan (shown in red) leads from the start to the goal states, while the local plan (green) guides the robot along this plan.

paths. The Local Planner is able to accurately track the global plan even through the aerial phase which requires body accelerations of over 6g and a maximum velocity of 2.55 m/s (Fig. 16). The Robot Driver is able to accurately produce the desired ground reaction forces (Fig. 17). In addition, both the visualizations and data logs shown in Figs. 15-17 were generated directly from the processing scripts that accompany Quad-SDK.

3.3.2 Large Rough Terrain Environment

This task tests the ability of Quad-SDK to reach a goal state extremely far away and over continuously rough terrain. The terrain covers an area measuring 75x75 body lengths (30x30 m) and contains large obstacles and holes distributed throughout. The terrain also contains random elevation noise with amplitude up to half the leg length of the robot (± 20 cm) and characteristic length as short as twice the body length. The required goal state is located 67 body lengths (26.9 m) away and behind several of the obstacles.

By leveraging the action parameterization which permits long-range connections of states, the

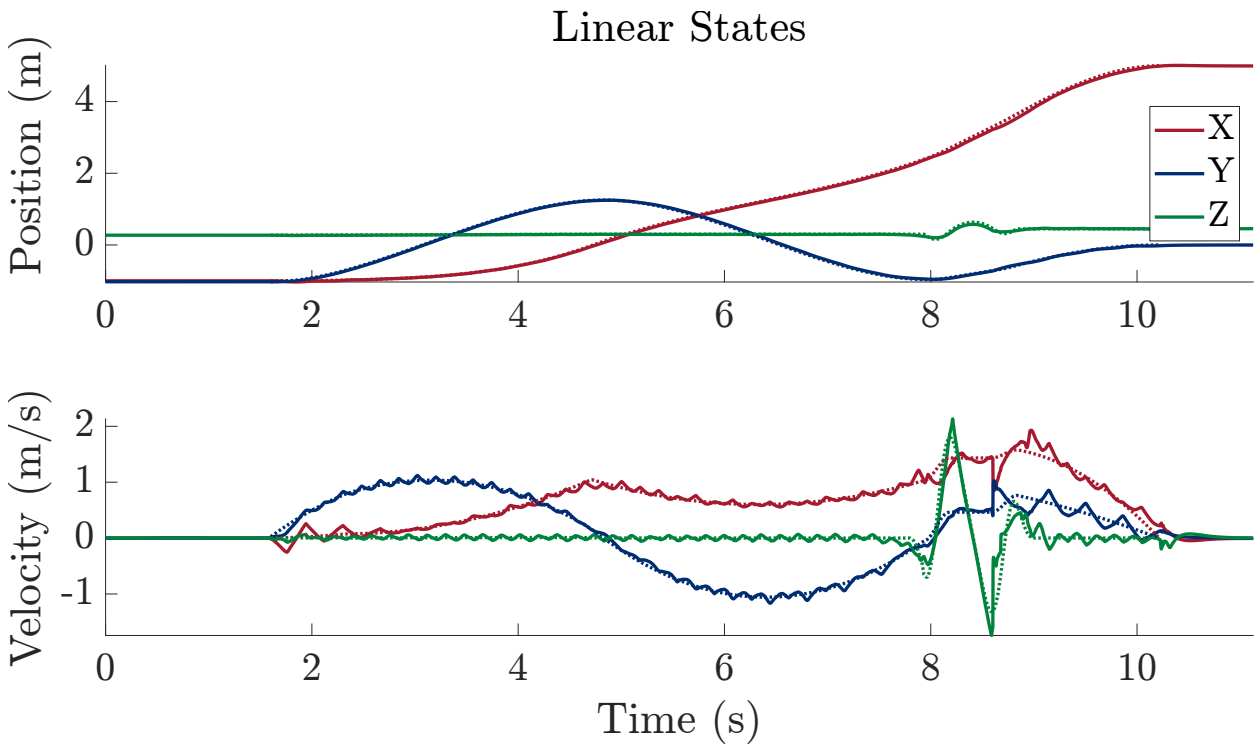


Figure 16: Local Minima and Leap Environment – Local Planner. The Local Planner yields state trajectories (solid) that closely track the global plan (dashed) over the unstructured terrain.

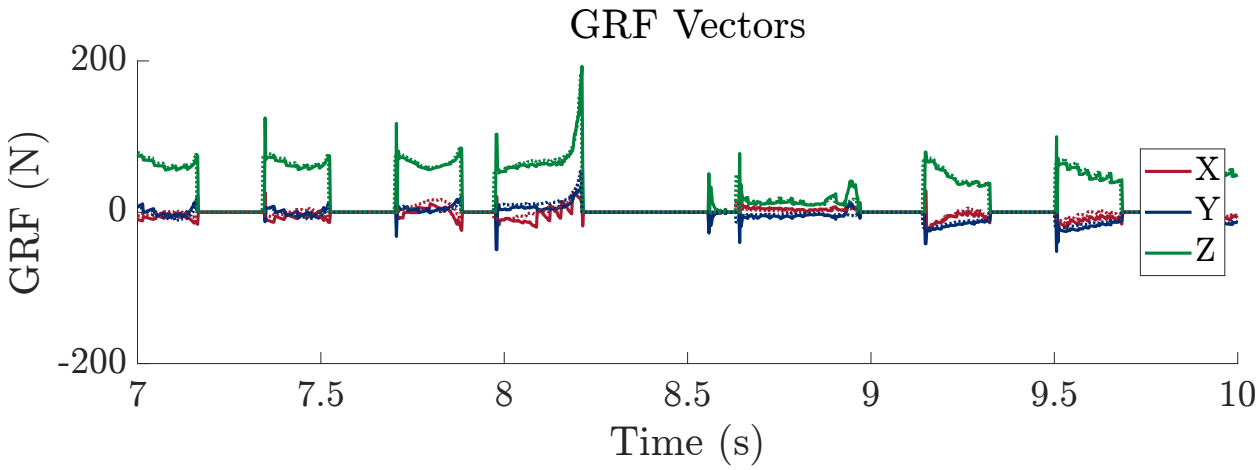


Figure 17: Local Minima and Leap Environment – Robot Driver. The Robot Driver accurately produces GRFs (solid) that closely track those requested by the Local Planner (dashed). Data shown is for the right rear leg while executing the leap.

Global Planner finds a feasible plan to the goal (shown in Fig. 18) in just 17 milliseconds. This plan measures 27.6 m in length, which is only 2.5 % longer than the (likely infeasible) straight line path between the start and the goal. The Local Planner is able to execute the entirety of this plan

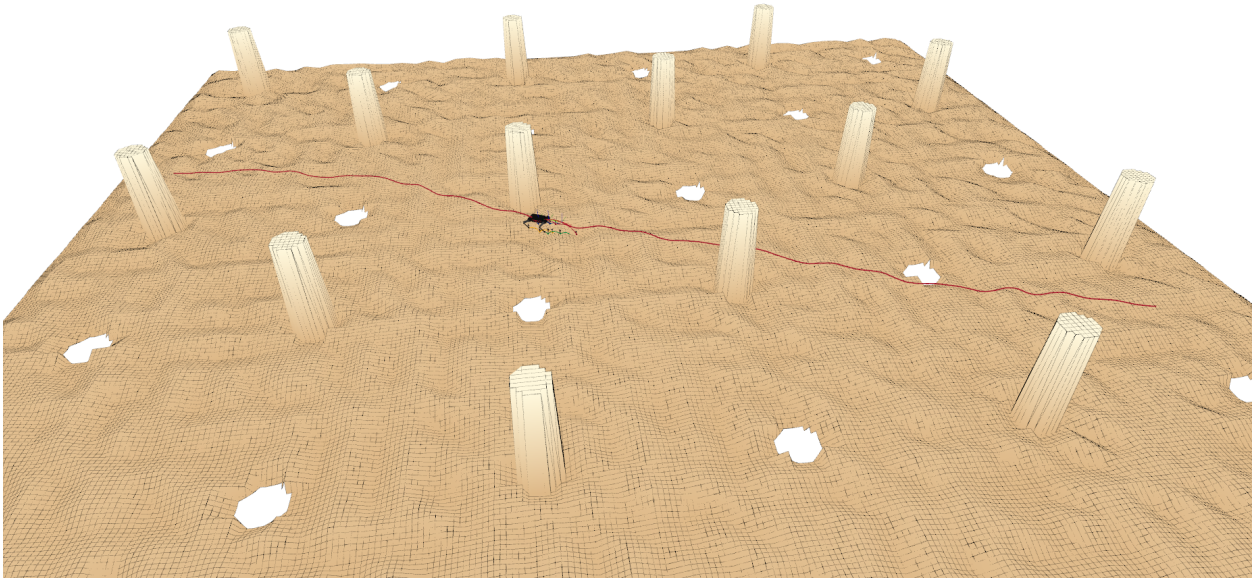


Figure 18: Large Rough Terrain Environment – Global Planner. The Global Planner can handle extremely-long horizon navigation tasks due to its efficient action parameterization.

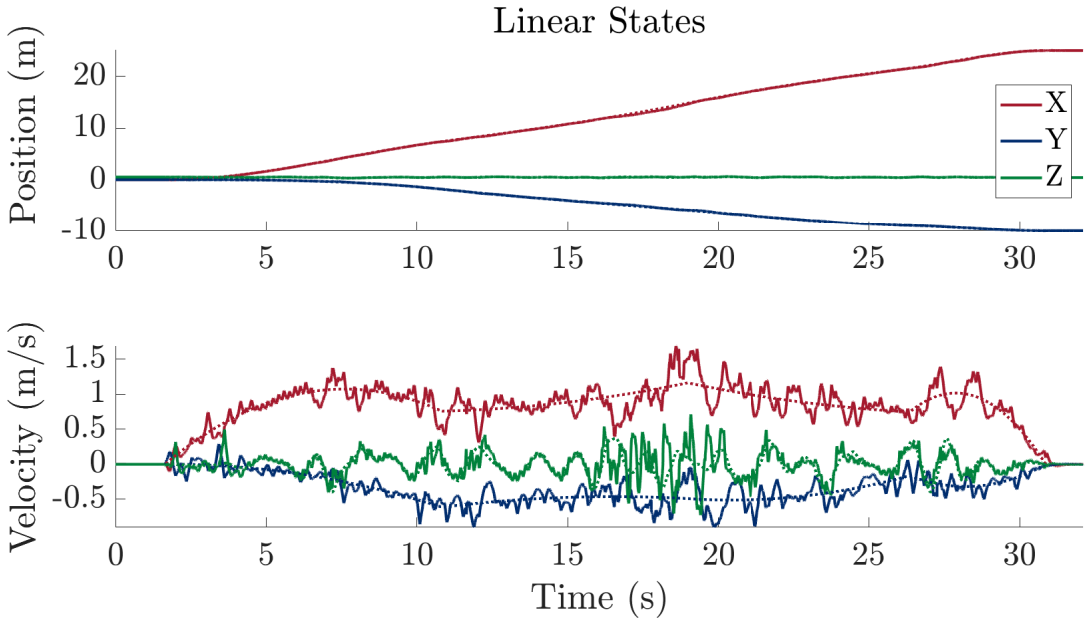


Figure 19: Large Rough Terrain Environment – Local Planner. The Local Planner handles the terrain roughness and contact scheduling even at high velocities so that the Global Planner can focus on finding efficient routes to the goal.

in 29.2 s for an average velocity of 2.4 body lengths per second (0.94 m/s).

3.3.3 Robustness to Sensor Error

In each of the prior tests the robot was provided with ground truth knowledge of the state and terrain map to isolate the performance of the planning and control algorithms. However during real-world deployment ground truth data is never available, and robots much be able to handle sensor data that is inaccurate. To test the robustness of the Quad-SDK stack to inaccurate sensor data, this task requires the robot to walk forwards at 1.0 m/s over rough terrain consisting of elevation white noise with amplitude up to 0.25 times the leg length (± 10 cm). The difference between the terrain maps is illustrated in Fig. 20. Additionally, additive Gaussian white noise with $\sigma_p = 0.01$ m is applied to the estimated position and $\sigma_v = 0.2$ m/s to the estimated velocity in random but aligned directions.

The resulting performance is shown in Figs. 21-22. The robot is able to successfully track the plan despite the imperfect knowledge. Tracking performance is significantly worse than in the rough terrain locomotion shown in Fig. 19, but sufficient to maintain stability. This is due to the stable contact dynamics and rapid replanning capabilities which Quad-SDK enables. Since stance phase control is largely driven by desired forces rather than position tracking, any missed contacts are quickly resolved by the large vertical GRF the Local Planner requests. Additionally, since the planner constantly updates the predicted foot trajectories with current foot information, the system quickly reacts to the discrepancy in terrain height or foot slip, choosing to trust the proprioceptive foot information rather than the inaccurate terrain data. Figure 22 shows the foot velocities during the behavior, which demonstrate these principles – instances of missed contact are immediately followed by large spikes of downward velocity due to the stance controller, and despite numerous foot slips (shown as spikes in lateral velocity) the system is able to maintain stability.

3.3.4 Hardware Deployment

This example demonstrates the ability of Quad-SDK to perform agile behaviors in hardware. Figure 23 shows snapshots of a one body-length leap executed on a quadrupedal platform. The framework is able to seamlessly transition from a 1.0 m/s trot into the leap and back again, while sta-

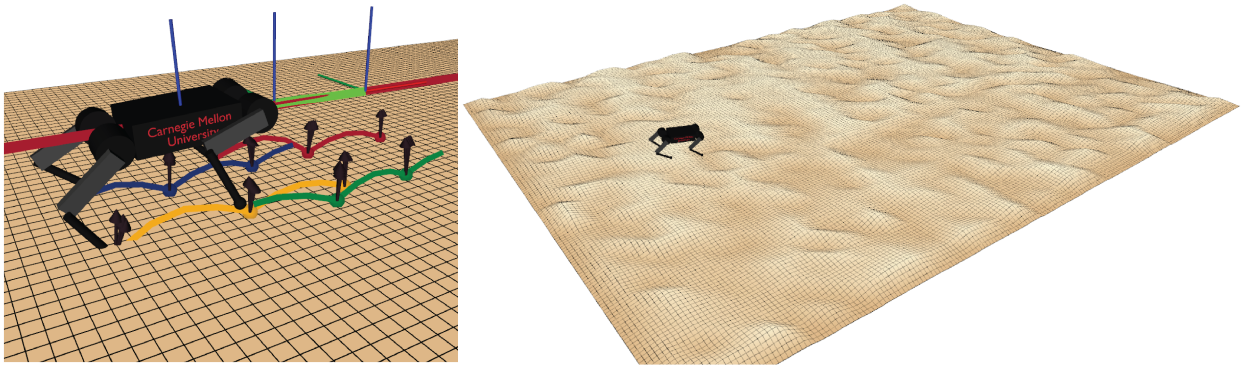


Figure 20: The robustness to sensor error test shows success despite imperfect knowledge of the system. Left – the terrain as perceived by the robot causes it to believe its foot is through the floor. Right – the actual terrain used in the simulation test.

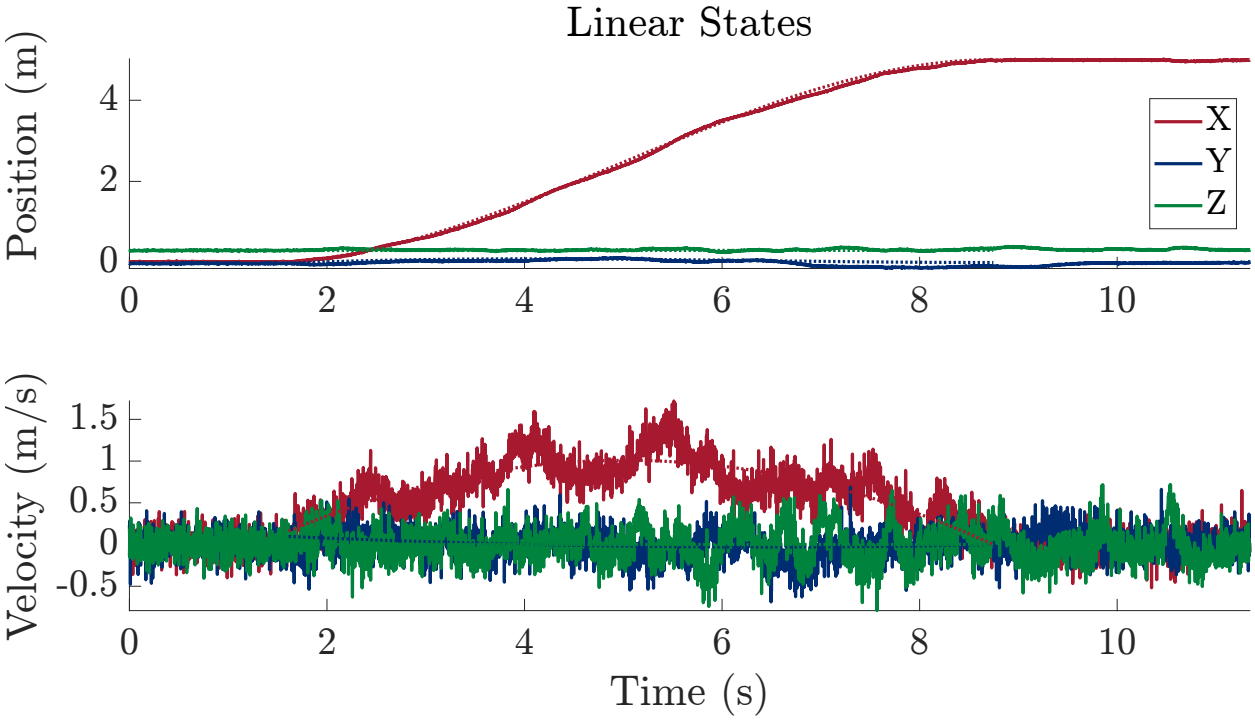


Figure 21: Despite the imperfect terrain and state knowledge, the stack provided by Quad-SDK is able to robustly execute the desired motion.

bilizing the behavior through a period of underactuation despite imperfect sensing and control. This performance demonstrates that the combination of the reduced-order model of the leaping behavior coupled with a two-gait cycle horizon MPC running over 100 Hz is sufficient to stabilize this highly dynamic behavior, suggesting that high-dimensional trajectory optimizations are not required to autonomously perform these motions.

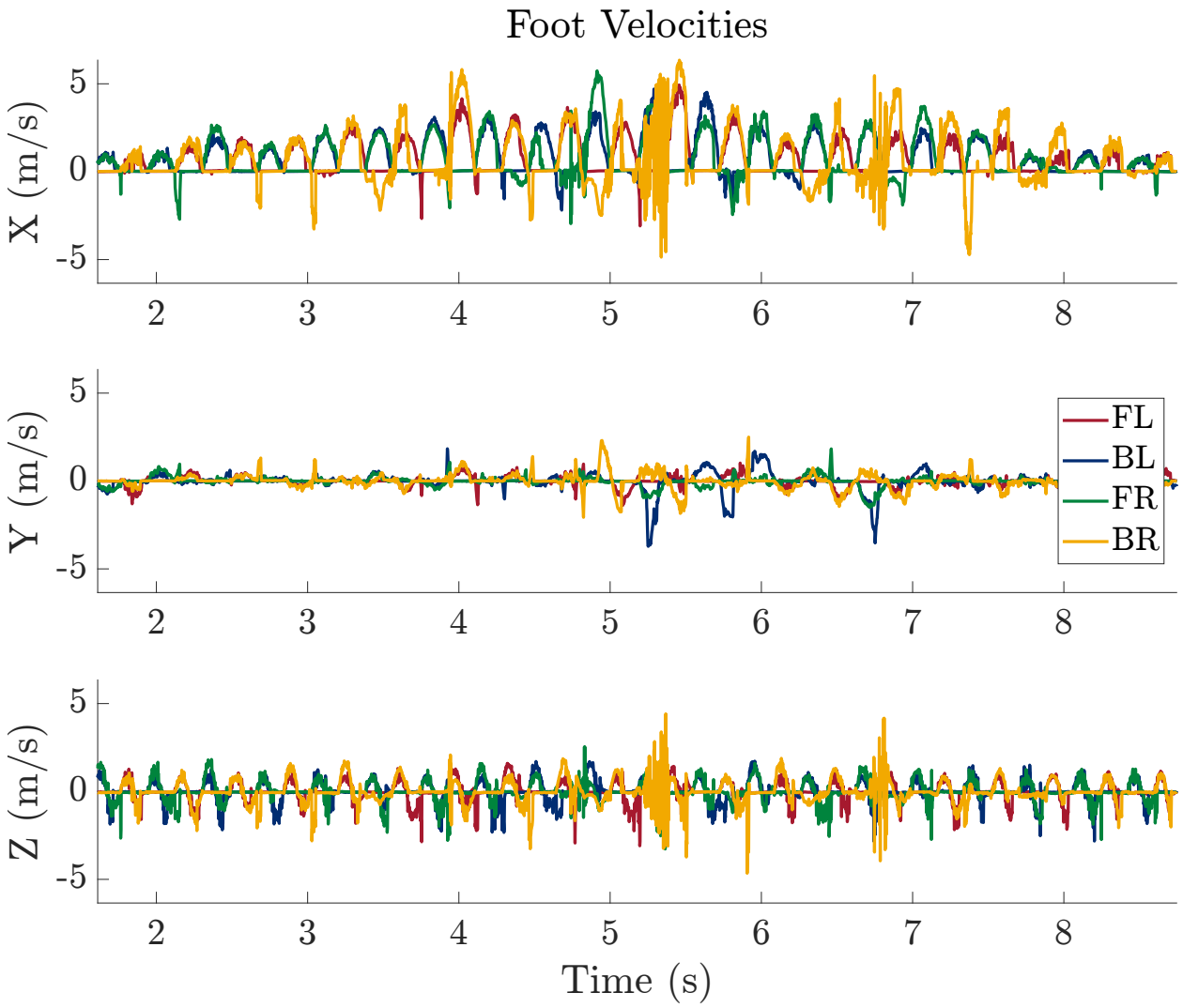


Figure 22: The foot velocities over time show the foot slip and height mismatch caused by sensor inaccuracies. The algorithms in Quad-SDK are able to stabilize the dynamic motion despite these disturbances.

3.3.5 Multi-robot Support

The examples in Fig. 24 demonstrate multi-robot support for multiple platforms. The robots are able to independently plan and execute long-horizon trajectories within a shared environment. Future releases will support multi-robot coordination that allows real-time inter-agent collision avoidance.

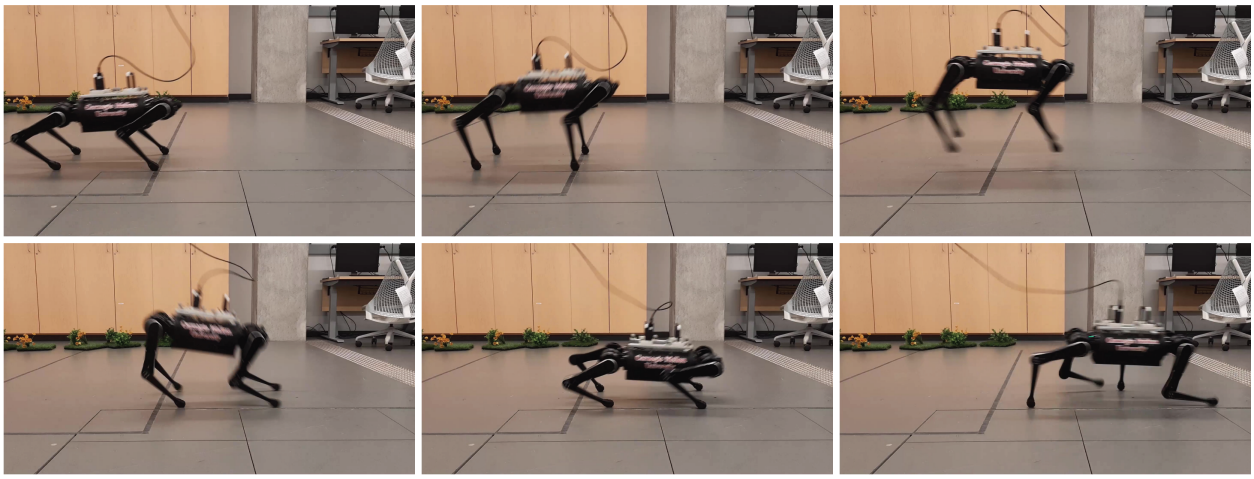
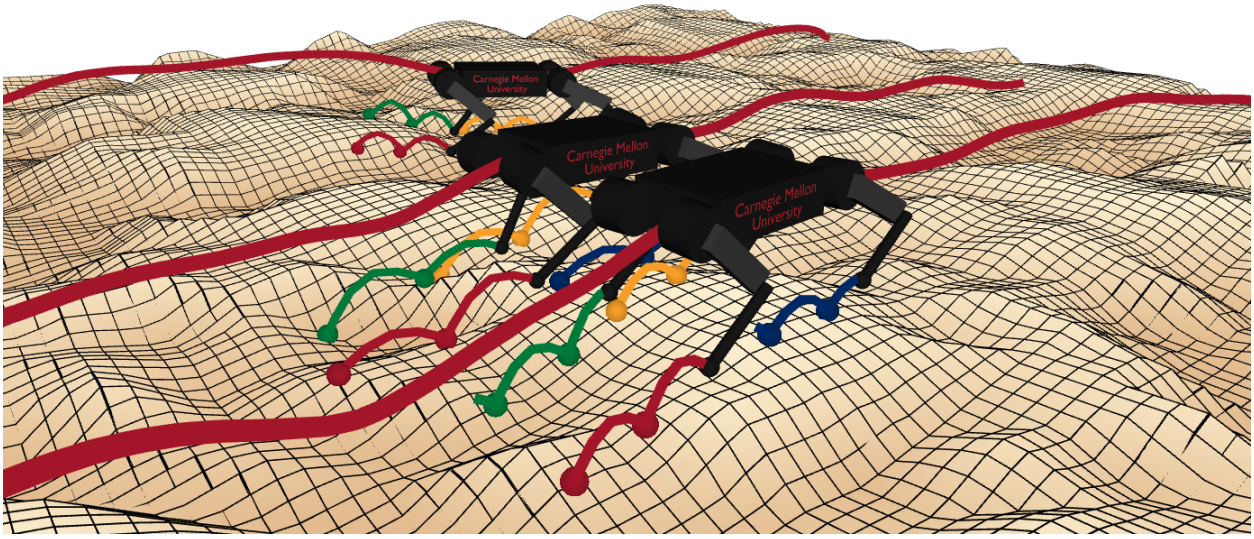


Figure 23: Quad-SDK executing a one body-length running leap on a hardware platform.

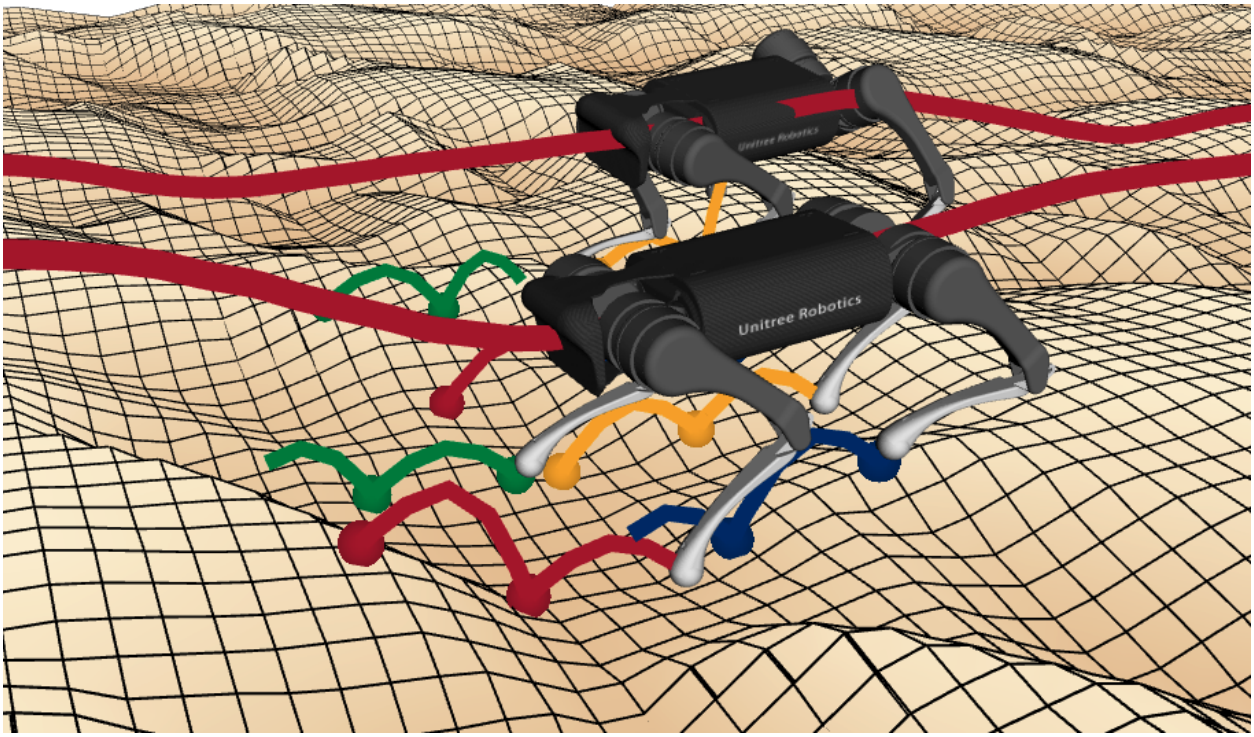
3.4 Conclusion

This section presents a high-level overview of Quad-SDK, an open source ROS-based full-stack software framework for agile quadrupedal locomotion. The package provides an extensible framework for developers to focus on high-level autonomy while enabling changes to low-level implementation of planning, control, estimation, and simulation components. The validity of the stack and its core features were demonstrated through experiments highlighting agile autonomy of multiple platforms for single and multi-robot scenarios. Future releases of the software will include onboard state estimation and perception-based terrain estimation for outdoor operation, as well as Python bindings and PyBullet simulation support for developing reinforcement learning algorithms.

Although this software is released via an open source license to promote collaboration within the community, we urge developers to consider the ethical impacts of their work in such a nascent field. We do not condone the use of this software in any use-of-force applications, as we believe quadrupeds – like all robots – should aid humans rather than harm them.



(a) Ghost Robotics Spirit 40



(b) Unitree Robotics A1

Figure 24: Quad-SDK supports multi-agent unstructured locomotion for multiple quadrupedal platforms, with independent planning and control stacks for each robot.

4 Adaptive Complexity Model Predictive Control

4.1 Introduction

As demand for robotic systems increases in industries like environmental monitoring, industrial inspection, disaster recovery, and material handling [Bellicoso et al., 2018; Hutter et al., 2017; Kolvenbach et al., 2020], so too has the need for motion planning and control algorithms that efficiently handle the complexity of their dynamics and constraints. Legged systems in particular are well suited for these applications due to their ability to traverse unstructured terrains with behaviors such as that shown in Fig. 25, yet they are so far largely restricted to conservative behaviors due to this complexity. A common approach to overcome these challenges is to break up the problem into a hierarchy of sub-problems which reason over progressively shorter horizons with increasing model complexity. This hierarchy improves computational efficiency which can be used to detect obstacles further away, react more quickly to disturbances, or reduce energy costs. However, this hierarchy is vulnerable to failures caused by omitting portions of the underlying model, raising a fundamental question of how to balance model fidelity and computational efficiency.

Inspiration for answering this question can be drawn from other scientific fields, in particular behavioral economics and neuromechanics. The famous "Thinking: Fast and Slow" framework theorizes that human cognitive function can be described with two systems which respectively handle rapid, simple processing and slow, deliberative reasoning such that "[the complex, slow system] is activated when an event is detected that violates the model of the world that [the simple, fast system] maintains" [Kahneman, 2011]. Extending this concept to the field of motion planning yields meta-planning methods which change their structure to leverage simple, fast models where possible and complex, slow ones where the simple model is inaccurate [Fridovich-Keil et al., 2018; Gochev et al., 2011]. However, it is not well understood under what exact conditions a given dynamical system may leverage a simple model without sacrificing stability and feasibility, nor is it clear when a more complex model should be used without making the computational overhead intractable.

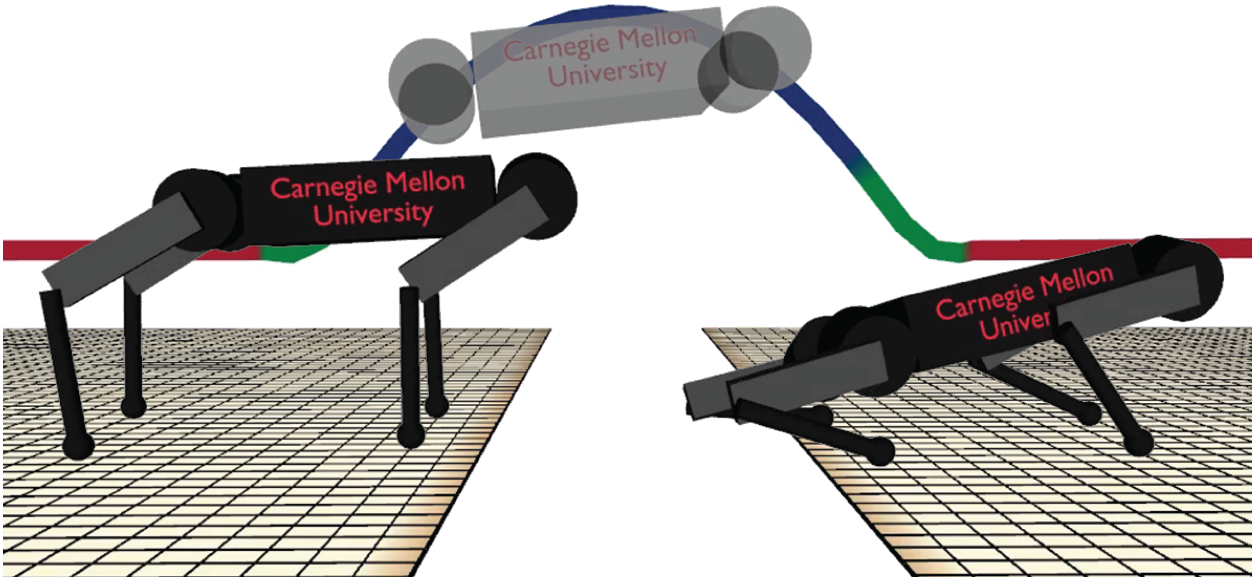


Figure 25: Adaptive complexity model predictive control selectively simplifies the model to promote efficiency without sacrificing stability. For example, during a legged leaping task joint information may be required during takeoff and landing but can be omitted elsewhere without affecting the behavior.

To investigate these questions, we employ the templates and anchors approach for analyzing model hierarchies [Full and Koditschek, 1999]. This framework describes relations between dynamical systems where complex behaviors (anchors) can be captured by simpler models (templates), and its connections to legged locomotion are well-studied in both animals and robots. From these observations we draw three hypotheses:

1. From the templates and anchors relationship we can derive sufficient conditions which identify regions of a behavior where complex dynamics and constraints can be safely simplified without compromising stability and feasibility (i.e. regions where the robot is following the template model).

2. Legged systems often operate in environments where these conditions are satisfied during the majority of behaviors and therefore stability and feasibility guarantees can often be retained even with simplified models.

3. During behaviors in which these conditions are not met, a controller which leverages adap-

tive complexity online can improve performance over fixed-complexity formulations by enabling more efficient motion planning while retaining stability guarantees.

We evaluate these hypotheses by constructing a formulation of model predictive control (MPC) that actively adapts the model complexity to the task. This is achieved by iteratively identifying regions of the horizon where the behavior can be safely expressed with a simpler model. This formulation is most efficient when the two models satisfy a relationship known as "exact anchoring" [Libby et al., 2016b], and in the worst case converges to the standard all-complex MPC formulation. We evaluate Hypothesis 1 by showing that this algorithm provides formal stability and feasibility properties with respect to the complex system. We evaluate Hypotheses 2 and 3 by applying this algorithm to a quadrupedal system and conducting simulation experiments on common environments which legged platforms may encounter. These results show that the majority of the behaviors in these environments admit feasible simplifications and that the resulting improvement in computational efficiency enables an increase in top speed and decrease in task duration compared to a system without these reductions. We also show that retaining knowledge of the dynamics and constraints in the complex system expands the range of executable tasks compared to a system which uniformly applies these reductions. In the particular case of legged leaping, this approach enables receding horizon execution of a body-length leaping behavior while considering joint constraints, which expands the leaping ability over prior methods which do not consider these constraints.

The organization of this paper is as follows: Section 4.2 covers related work in greater detail, and Section 4.3 formulates the problem and introduces notation. Section 4.4 provides an overview of the algorithm, whose formal properties are explored in Section 4.5. This algorithm is applied to legged locomotion models in Section 4.6, and performance for these systems is quantified in Section 4.7. Section 4.8 discusses limitations and future extensions enabled by this work, followed by a short Appendix in Section 4.9 with some proofs of the theories presented.

4.2 Related Work

Dynamic motion planning and control for systems with intermittent contact is inherently difficult. However, enabling agile autonomy for such systems is critical for real-world applications that require the robot to touch the world. In particular, legged robots have significant potential for real-world deployment. But, they often suffer from difficulties arising from hybrid dynamics, high state dimensionality, and non-convex constraints on their kinematics and dynamics. Such problems render even basic motion planning problems PSPACE-complete [Reif, 1979], and as a result existing algorithms to solve them globally for dynamic legged systems cannot operate in real-time [Dai et al., 2014; Hauser et al., 2008; Mombaur, 2009; Posa et al., 2014].

In general, the most common approach used to address the planning and control challenges of legged locomotion has been through leveraging some kind of model reduction, wherein the problem is solved with a reduced subset of the state and dynamics. The solution of this simpler problem is then passed to another system with a more complex model which fills in the additional details. The hierarchical nature of this approach enables optimization of each algorithm based on salient features of the problem such as the timescales of the dynamics or rates of sensor information. Examples of this approach include efficient global planners focused on exploration [Bartoszyk et al., 2017; Fernbach et al., 2017; Norby and Johnson, 2020; Tonneau et al., 2018] to local planners that plan contact phases over a few gait cycles [Winkler et al., 2018a] to whole-body controllers with full-order representations over very small horizons [Kuindersma et al., 2016; Neunert et al., 2018; Sentis and Khatib, 2006]. While these hierarchies have been shown to be capable of producing dynamic locomotion, they face a fundamental problem in that dynamics or constraints in the omitted space can render the desired motion sub-optimal or even infeasible. Conservative assumptions in the simple model may fail to produce solutions in difficult environments, and optimistic assumptions may lead to infeasible behaviors that the more myopic complex system may not recognize until too late.

Several methods have been explored to resolve the interface between these layers. One straightforward approach is to use the complex model to assess the simple solution, either by providing a

boolean feasible/infeasible classification, computing some value function, or indicating new search directions [Carpentier et al., 2017; McConachie et al., 2020; Plaku et al., 2010; Zucker et al., 2011]. This can be efficient since checking a solution in a higher-order space is easier than searching for one, although it does not allow the simple model to directly reason about the dynamics or constraints in the complex space.

Another promising way to resolve this is to employ an adaptive planning framework to reason over different models based on the task and constraints. One flavor of this approach plans over a mixture of models of varying degrees of fidelity with some pre-defined rules that guide when to use each and how to transition between them [Brandao et al., 2019; Kapadia et al., 2013; Norby and Johnson, 2020]. Other approaches leverage an adaptive composition of models with different safety bounds to trade between performance and robustness or expand the problem dimensionality as needed to find collision-free paths. [Dornbush et al., 2018; Fridovich-Keil et al., 2018; Gochev et al., 2011; Styler and Simmons, 2017; Zhang et al., 2012]. Our approach is more similar to the latter in terms of the underlying adaptation mechanism, but differs in that we derive the exact conditions under which transitioning between models of varying fidelity can be done without sacrificing stability and feasibility. Furthermore, we demonstrate how such a mechanism can be applied to a receding horizon framework for online planning and control.

Similar planning and control problems can also be solved online using receding horizon methods. In particular, model predictive control (MPC) is an iterative receding-horizon optimization framework that has been commonly used to solve constrained optimal control problems [Allgöwer and Zheng, 2012]. In the context of dynamic legged locomotion, MPC often computes feasible body and/or joint trajectories in order to track a higher level reference plan while respecting dynamic, state, and control constraints. Works such as [Di Carlo et al., 2018; Laurenzi et al., 2018; Shi et al., 2019] compute the desired ground reaction forces using a single rigid body model, which are realized as joint torques using a whole-body controller. Although computationally efficient, this approach typically uses a single simplified model under the assumption that motions of the legs have minimal effects on the body. This can be limiting in performing more aggressive motions

such as those that require flight phases or on robots with high "Centroidal Inertia Isotropy" [**?**].

While simplified model MPC approaches assume a representative reduced-order model (template) exists that fairly approximates the full-order model (anchor), they often do not examine the validity of this approximation. Some approaches have studied safe controller synthesis for the template model while ensuring constraint satisfaction of the anchor model through bounding the differences of the two models using reachability analysis [Liu et al., 2020], approximate simulation properties [Kurtz et al., 2019], or learning any unmodeled differences [**?**]. Another approach uses pre-defined ratios to mix the complex model for the immediate future with the simple model for longer horizon planning, which results in more robust locomotion compared to fixed-complexity formulations [Li et al., 2021]. Our approach takes inspiration from these in that it defines the exact conditions under which the higher order model can be simplified without violation of formal guarantees, but critically differs in that we adaptively mix models of varying fidelity within any planning horizon based on local feasibility. This allows us to leverage the fidelity of the complex model when necessary while taking advantage of the computation speed enabled by the simple model for planning longer horizon motions.

Many of these hierarchical planning and control approaches have been shown to effectively perform agile and dynamic motions in simulation and hardware. However, they often struggle in generating and executing motions that require the robot to operate at its kinematic and dynamic limits. These motions are critical in enabling behaviours like stepping and leaping over gaps, stairs, and non-traversable obstacles, which are essential in navigating unstructured terrains. Previous approaches have relied on executing trajectories that have been optimized offline or prior to execution, lack longer horizon planning, or do not consider joint kinematics or constraints [Johnson and Koditschek, 2013a; Kolvenbach et al., 2019; Nguyen et al., 2019; Ponton et al., 2021; **?**]. Our approach allows the robot to plan for these agile behaviours in a receding-horizon manner while reasoning about constraints over a significantly long horizon, which is shown to expand leaping capabilities in simulation.

4.3 Preliminaries

To clarify the operation of adaptive complexity MPC and its properties, we define a formulation for model predictive control and the closed-loop system it yields following [Rawlings et al., 2017]. Consider a nonlinear, discrete-time, dynamical system which evolves on state manifold X under admissible controls $\mathcal{U}$ and with dynamics f,

$$x_{k+1} = f(x_k, u_k) \tag{26}$$

where $x_k \in X, u_k \in \mathcal{U}$ are the current state and control at time k, and $x_{k+1} \in X$ is the successor state. Let $\mathcal{X}$ be the set of all feasible states within manifold X. Let $z_k \coloneqq (x_k, u_k)$ define a state-control pair such that $x_{k+1} = f(z_k)$, and let feasible states and controls be defined by the set $\mathcal{Z} = \mathcal{X} \times \mathcal{U}$. Let the set $\mathcal{X}_N$ denote the basin of attraction of the controller parameterized by N. We list a few standard assumptions on the system in (26) and the set $\mathcal{X}_N$:

Assumption 1. (A) $f(0, 0) = 0$ (the origin is an equilibrium point).
(B) $\exists u \in \mathcal{U} \mid f(x, u) \in \mathcal{X}_N \; \forall x \in \mathcal{X}_N$ ($\mathcal{X}_N$ is control positive invariant).
(C) $\mathcal{X}$ and $\mathcal{U}$ are compact and contain the origin in their interiors.

To formalize model predictive control, first define a predicted control trajectory with horizon N as $\mathbf{u} = [u_0, u_1, \ldots, u_{N-1}]$ and a predicted state trajectory as $\mathbf{x} = [x_0, x_1, \ldots, x_N]$. For simplicity we define the resulting predicted state-control pair trajectory as $\mathbf{z} = [z_0, z_1, \ldots, z_{N-1}]$ such that $z_i = (x_i, u_i)$, where i denotes the index within the horizon defined at time k. The optimal control problem (OCP) solved in the standard NMPC formulation with terminal cost and region is thus

$\mathcal{P}(x_k)$,

$$\mathcal{P}(x_k): \quad V_N^*(x_k) = \min_{\mathbf{u}} \sum_{i=0}^{N-1} L(z_i) + V_t(x_N) \tag{27a}$$

$$\text{s.t.} \quad x_{i+1} = f(z_i)\ \forall i = 0, \ldots, N-1 \tag{27b}$$

$$z_i \in \mathcal{Z}\ \forall i = 0, \ldots, N-1 \tag{27c}$$

$$x_0 = x_k \tag{27d}$$

$$x_N \in \mathcal{X}_t \tag{27e}$$

where $L(\cdot), V_t(\cdot)$ are the stage and terminal costs, $\mathcal{Z}$ is the set of feasible state-control pairs, and $\mathcal{X}_t$ is a terminal set. Let $\mathbf{u}^*$ be the control trajectory corresponding to the optimal solution of (27). The control law defined by NMPC is determined by solving (27) at each time k and applying the first control, i.e. $u_k = u_{0|_k}^*$. This defines the state feedback policy $h(x_k)$ and resulting closed loop system $f_h(x_k)$,

$$h(x_k) \coloneqq u_{0|_k}^* \tag{28}$$

$$f_h(x_k) \coloneqq f(x_k, h(x_k)) \tag{29}$$

Standard stability proofs for NMPC formulations typically rely on showing that the closed-loop system admits a Lyapunov function which is upper and lower bounded by strictly increasing functions of state and is strictly decreasing in time (for asymptotic stability) or bounded in magnitude by the control input (for Input-to-State Stability). We borrow the standard definitions of strictly increasing functions $\mathcal{K}$ and $\mathcal{K}_\infty$ as well as Lyapunov functions and asymptotic stability from [Limon et al., 2009].

For the closed-loop system in (29) to yield provable stability, assume the following properties on $L(\cdot), V_t(\cdot)$, and $\mathcal{X}_t$.

Assumption 2. (A) $\exists \alpha_L, \alpha_U, \alpha_{L,f}, \alpha_{U,f} \in \mathcal{K}_\infty \mid \alpha_U(|x|) \leq L(x,u) \leq \alpha_L(|x|) \forall x \in \mathcal{X}, u \in \mathcal{U}$ and $\alpha_{U,t}(|x|) \leq V_t(x) \leq \alpha_{L,t}(|x|) \forall x \in \mathcal{X}_t$ (stage and terminal cost are upper and lower bounded by

$\mathcal{K}_\infty$ functions).

(B) A solution to (27) exists for all $x_k \in \mathcal{X}_N$.

(C) The functions $L(x,u), V_t(x)$, and f are all twice differentiable with respect to x and u.

(D) $\exists \alpha_{V_t} \in \mathcal{K}_\infty, h_t(x) \mid V_t(f(x,h_t(x))) - V_t(x) \leq -\alpha_{V_t}(|x|)\ \forall x \in \mathcal{X}_t$ (terminal control law decreases cost).

(E) The control law defined in (28) satisfies Assumption 1A.

It is a known result that under these conditions, the system defined in (29) is asymptotically stable for all $x_0 \in \mathcal{X}_N$ [Allgöwer and Zheng, 2012; Limon et al., 2009].

4.4 Adaptive Complexity MPC

4.4.1 Algorithm Overview

The core idea of adaptive complexity is to leverage models of differing complexity to simplify the model in regions where feasibility is assured and only increase complexity as needed to guarantee feasibility and stability properties. Our approach is to define a simplicity set S_k at each time k which indicates whether state-action pairs at a given time in the horizon after k can be simplified, and piece-wise dynamics that can propagate the state both into and out of this set. Conditions on this set S_k can be directly drawn from the literature on templates and anchors [Full and Koditschek, 1999] – elements of S_k represent times corresponding to state-action pairs that follow a feasible, attracting, invariant submanifold within the complex ("anchor") space, i.e. follow the "template" dynamics. In other words, membership in S_k implies that the system remains on the manifold after applying the complex dynamics and without violating constraints. This knowledge allows the system to optimize directly on the manifold in these regions, omitting the lifted component of the system and thus improving efficiency, while retaining these components when the feasibility or invariant properties are no longer satisfied. Note that while this work assumes one template per anchor, it could be further extended to include multiple templates describing different reductions of the anchor dynamics.

4.4.2 Complex and Simple System Definitions

Let the original system defined in (26) be "complex" which is clarified by the superscript $(\cdot)^c$. Let the "simple" system be denoted with the superscript $(\cdot)^s$ such that the state x^s lies on the manifold X^s, where $\dim X^s < \dim X^c$. These states are related by the state reduction $\psi_x : X^c \to X^s$ defined as $x^s = \psi_x(x^c)$. Let each system have controls u^c and u^s defined over manifolds U^c and U^s which are related by the control reduction $\psi_u : U^c \times X^c \to U^s$ defined as $u^s = \psi_u(u^c, x^c)$. Define the state-control pairs as $z^c = (x^c, u^c)$ and $z^s = (x^s, u^s)$ which lie on manifolds $Z^c \coloneqq X^c \times U^c$ and $Z^s \coloneqq X^s \times U^s$. This permits the definition of the reduction $\psi : Z^c \to Z^s$ defined as $z^s = \psi(z^c) = (\psi_x(x^c), \psi_u(u^c, x^c))$. Note that these many-to-one projection defines which components of the complex system are retained in the simple system.

Define the mapping $\psi^\dagger$ such that $\psi \circ \psi^\dagger = I$, where I is the identity matrix. Let $\psi_x^\dagger$ and $\psi_u^\dagger$ give the outputs of $\psi^\dagger$ corresponding to state and control variables respectively. This choice defines a particular heuristic – among many possible operators – for how states and controls in the null-space of $\psi_{(\cdot)}$ should correspond to the simple system. For stability purposes we choose that this maps to the origin (i.e. the reference) for variables contained in the null space of ψ.

The dynamics and constraints for the complex system have already been defined in Section 4.3. Define the dynamics f^s and constraints of the simple system $\mathcal{Z}^s$,

$$x_{k+1}^s = f^s(z_k^s) \tag{30}$$

$$z_k^s \in \mathcal{Z}^s. \tag{31}$$

Let the basin of attraction of the complex system be denoted $\mathcal{X}_{N^c}$ under horizon length N^c.

4.4.3 Adaptive System Definition

We seek an adaptive control law which leverages the simple system when the system can feasibly remain on the manifold Z^s and the complex system when it cannot. Define another set of state and control variables x_i^a, u_i^a, and z_i^a which represent the adaptive mixed system which is used to solve

the OCP. We use S_k to assign these quantities at a time i in the horizon to a particular manifold,

$$z_i^a = \begin{cases} z_i^c \in Z^c, & i \notin S_k \\ z_i^s \in Z^s, & i \in S_k \end{cases} \tag{32}$$

The adaptive states and controls can be either lifted to the complex manifold $z^l \in Z^c$ or reduced (projected) to the simple manifold $z^r \in Z^s$ by leveraging ψ and $\psi^\dagger$ at time i,

$$z_i^l = \begin{cases} z_i^a, & i \notin S_k \\ \psi^\dagger(z_i^a), & i \in S_k \end{cases} \tag{33}$$

$$z_i^r = \begin{cases} \psi(z_i^a), & i \notin S_k \\ z_i^a, & i \in S_k \end{cases} \tag{34}$$

Next we define the dynamics of the adaptive system which are used to solve the OCP,

$$x_{i+1}^a = f^a(z_i^a) \coloneqq \begin{cases} f^c(z_i^a) & i, i+1 \notin S_k \\ \psi \circ f^c(z_i^a) & i \notin S_k,\, i+1 \in S_k \\ f^c \circ \psi^\dagger(z_i^a) & i \in S_k,\, i+1 \notin S_k \\ f^s(z_i^a) & i, i+1 \in S_k \end{cases} \tag{35}$$

where x_{i+1}^a is the successor state in the adaptive system. The OCP for the adaptive system uses these dynamics to construct feasible motions over a prediction horizon $N^a \geq N^c$.

Denote a predicted adaptive control sequence over horizon N^a as $\mathbf{u}^a = [u_0^a, u_1^a, \ldots, u_{N^a-1}^a]$, a predicted adaptive state sequence as $\mathbf{x}^a = [x_0^a, x_1^a, \ldots, x_{N^a}^a]$, and a predicted adaptive state-control pair sequence as $\mathbf{z}^a = [z_0^a, z_1^a, \ldots, z_{N^a-1}^a]$. Let the lifted and reduced forms of these trajectories be

denoted as $\mathbf{z}^l$ and $\mathbf{z}^r$. We define the constraints $\mathcal{Z}_i^a$ in the adaptive system,

$$\mathcal{Z}_i^a \coloneqq \begin{cases} \mathcal{Z}^c & i \notin S_k \\ \mathcal{Z}^s & i \in S_k \end{cases}. \tag{36}$$

With these definitions in place we now state the OCP $\mathcal{P}^a(x_k^c)$ which is solved to determine the control input u_k^c,

$$\mathcal{P}^a(x_k^c): \quad V_{N^a}^{*a}(x_k^c) = \min_{\mathbf{u}^a} \sum_{i=0}^{N^a-1} L^a(z_i^a) + V_t(x_{N^a}^a) \tag{37a}$$

$$\text{s.t.} \quad x_{i+1}^a = f^a(z_i^a)\ \forall i = 0, \ldots, N^a - 1 \tag{37b}$$

$$z_i^a \in Z_i^a\ \forall i = 0, \ldots, N^a - 1 \tag{37c}$$

$$x_0^a = x_k^c \tag{37d}$$

$$x_{N^a}^a \in \mathcal{X}_t^c \tag{37e}$$

where the adaptive cost function is equal to the complex system cost evaluated on the lifted state-control pair,

$$L^a(z_i^a) = \begin{cases} L^c(z_i^a) & i \notin S_k \\ L^c(\psi^\dagger(z_i^a)) & i \in S_k \end{cases} \tag{38}$$

Let $\mathcal{X}_{N^a}$ be the set of states for which the solution to (37) exists. Let the optimal value function found in (37) correspond to the control trajectory $\mathbf{u}^{*a} = [u_0^{*a}, u_1^{*a}, \ldots, u_{N^a-1}^{*a}]$ and corresponding lifted trajectory $\mathbf{u}^{*l}$. The control law defined by adaptive complexity MPC is determined by solving (37) at each time k and applying the first control, i.e. $u_k^c = u_{0|_k}^{*l}$. Define the state feedback policy

$h^a(x_k^c)$ and resulting closed loop system $f_{h^a}^c(x_k^c)$,

$$h^a(x_k^c) \coloneqq u_{0|_k}^{*l} \tag{39}$$

$$f_{h^a}^c(x_k^c) \coloneqq f^c(x_k^c, h^a(x_k^c)) \tag{40}$$

Note that each term in the OCP defined in (37) converges to its complex counterpart if $S_k = \{\}$, so the behavior of the original MPC closed-loop system defined in (26) can always be retained. However, we seek to find the minimal complexity required to still ensure stability of the closed-loop system.

4.4.4 Conditions on the Complexity Set

The set S_k clearly cannot be arbitrary in order to maintain stability, as ignoring some uncontrolled component of the system dynamics could easily cause undesired behavior. To avoid this we define a notion of the admissibility of S_k which is needed to show that the state and control trajectory in the adaptive space matches their realizations in the complex system.

Definition 1. (Admissibility of S_k) The simplicity set S_k defined at time k is admissible if the following conditions hold for all $i \in S_k$,

$$i \in 1, 2, \ldots, N^a - 1 \tag{41a}$$

$$\psi^\dagger \circ \psi(z_i^l) \in \mathcal{Z}^c \tag{41b}$$

$$\psi_x^\dagger \circ f^s \circ \psi(z_i^l) = f^c(z_i^l) \tag{41c}$$

$$\psi_x^\dagger \circ \psi \circ f^c(z_{i-1}^l) = f^c(z_{i-1}^l). \tag{41d}$$

The condition (41a) requires that the first and last state be complex, which ensures the predicted trajectory matches the actual dynamics and that the system is able to reach the (possibly complex) terminal state. The conditions (41b) require that the state and control on the manifold at i are feasible in the complex space, (41d) requires the dynamics from the prior state and control lead to

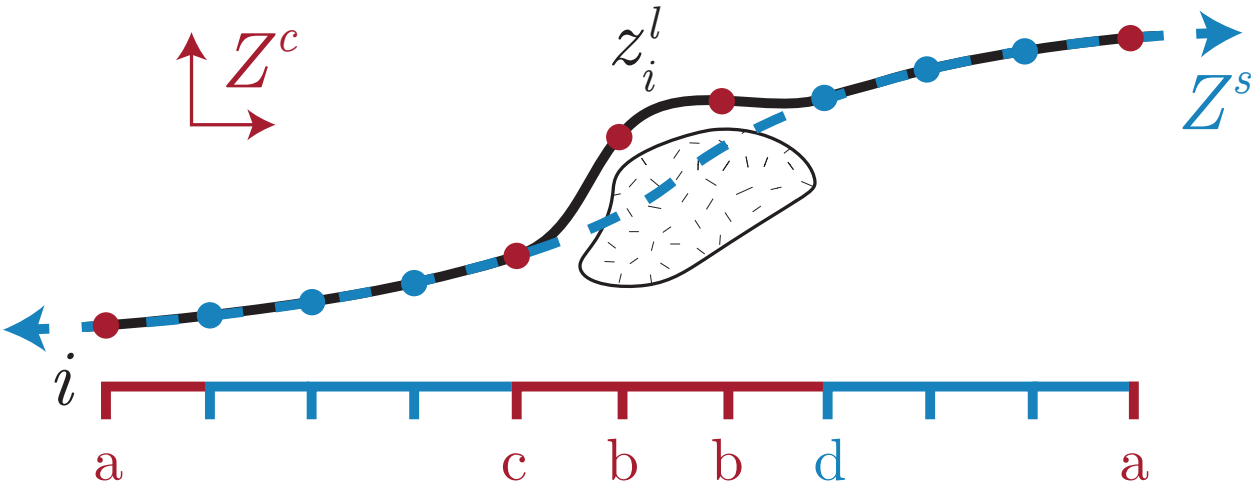


Figure 26: Elements in the horizon i are in the set S_k if they are feasible and stay on the manifold Z^s (illustrated as a blue dashed 1D curve). The adaptive system allows z_i^l to leave this submanifold while remaining in the manifold Z^c (the surrounding white 2D space). Elements in the set S_k are denoted in blue, elements not in this set are denoted in red and labeled with the condition of (41) they satisfy or violate.

the manifold, and (41c) requires that the dynamics applied to the current state yields a successor state on the manifold, i.e. the complex space "exactly anchors" the simple space (in the sense of [Libby et al., 2016b, Appendix A]). These concepts are illustrated in Fig. 26. To codify the admissibility of a simplicity set S_k, define the set of possible simplicity sets as $\mathcal{S}$, and the set of all admissible sets as $\mathcal{S}_a$. Membership in $\mathcal{S}_a$ can be determined by computing the lifted prediction trajectory $\mathbf{z}^l$ and checking the conditions in (41) for each $i \in S_k$. Also note that removing any element from an admissible simplicity set does not invalidate the admissibility property.

In order to show the stability of adaptive complexity MPC, we require that S_k be admissible for all k.

Assumption 3. $S_k \in \mathcal{S}_a \; \forall k \geq 0$.

The remaining challenge is ensuring the admissibility of S_k, which is inductively handled by Algorithm 1 and for which recursive admissibility is proven in Sec. 4.5.3. There are multiple methods to ensure the initial S_0 is admissible, such as initializing with $S_0 = \{\}$ which is guaranteed to be admissible, or iteratively solving (37) and updating S_0 until an admissible set is found. After

Algorithm 1 Adaptive Complexity Model Predictive Control

Given x_0^c, S_0, S^f, N^a
$k \leftarrow 0$
$S_0^a \leftarrow S_0$
repeat
  $S_k \leftarrow S_k^a \cap S^f$
  $\mathbf{z}^{*a} \leftarrow \mathcal{P}^a(x_k^c)|_{N^a, S_k}$ ▷ (37)
  $\mathbf{z}^{*l} \leftarrow \psi^\dagger(\mathbf{z}^{*a})$ ▷ (33)
  $u_k^c \leftarrow u_{0|_k}^{*l}$ ▷ (39)
  $x_{k+1}^c \leftarrow f^c(x_k^c, u_k^c)$ ▷ (40)
  $S_{k+1}^a \leftarrow \{i - 1 \mid i \in S_k^a \wedge z_i^l \text{ satisfies } (41)\}$
  $k \leftarrow k + 1$
until finished

an initial set is found, successor sets can be found by combining two simplicity sets, S_k^a and S^f. The set S_k^a adaptively identifies regions that can be safely simplified, while S^f requires that certain elements always remain complex for stability and admissibility, in particular the first and last element. After a solve, S_k^a is updated by checking (41) to determine which states can be simplified. Its elements are then shifted in time, i.e. $S_{k+1}^a = \{i - 1 \mid i \in S_k^a\}$, and combined with the fixed set to yield the successor simplicity set $S_{k+1} = S_{k+1}^a \cap S^f$. This approach is sufficient to guarantee admissibility in the nominal case with a perfect model, where the only new portion of the optimal trajectory is the last element which is always covered by S^f. Robustness can be improved at the expense of computational effort by expanding S^f to include more elements to ensure S_k remains admissible under disturbances. This is similar to the MPC formulation in [Li et al., 2021], but includes the adaptive term S_k^a to ensure feasibility across the entire horizon.

4.4.5 Formal Definition of Adaptive Complexity MPC Algorithm

With the control law we can now summarize adaptive complexity MPC as an iterative algorithm shown in Algorithm 1. This procedure combines the fixed and adaptive simplicity sets, solves the OCP, updates the adaptive simplicity set from its solution, then applies the first element of the control trajectory. Note that this algorithm is no different from standard MPC formulations with the exception of the definition of the simplicity set and lifting of the resulting trajectory.

4.5 Theoretical Analysis

This section describes the theoretical properties of adaptive complexity MPC. We first show that constraints of the original OCP in (27) are satisfied by solutions of the adaptive OCP in (37) under assumptions on the admissibility of S_k (Sec. 4.5.1). We use this result to show recursive feasibility of the adaptive OCP and thus asymptotic stability of the origin of the closed loop system (Sec. 4.5.2). We show that Algorithm 1 satisfies the assumption on admissibility of S_k (Sec. 4.5.3), and that the basin of attraction of the resulting system is no smaller than the original complex MPC system and possibly larger since the horizon length could be expanded with the additional computational capabilities (Sec. 4.5.4).

4.5.1 Optimal Control Problem Constraint Satisfaction

We begin by showing that admissibility of S_k results in a lifted trajectory which matches the solution to the closed-loop dynamics of the actual complex system, and therefore the constraints of the original OCP in (27) are satisfied by solutions of the adaptive OCP in (37). Let the solutions of the closed-loop dynamics starting at state x_k^c under a given control trajectory $\mathbf{u}^c$ for duration i be expressed by the function ϕ defined as $\phi(i, x_k^c, \mathbf{u}^c) \coloneqq x_{k+i}^c$.

Proposition 1. *Suppose Assumption 3 is satisfied. The predicted state at time i is equal to the solution to the complex dynamical system under the lifted predicted controls, i.e. $x_i^l = \phi(i, x_0^c, \mathbf{u}^l)$.*

Proof. We prove this by induction. For the base case $i = 0$, since $0 \notin S_k$, $x_0^l = x_0^c = \phi(0, x_0^c, \mathbf{u}^l)$. For the induction step we need to show that $x_i^l = \phi(i, x_0^c, \mathbf{u}^l)$ implies $x_{i+1}^l = \phi(i+1, x_0^c, \mathbf{u}^l)$. We obtain $\phi(i+1, x_0^c, \mathbf{u}^l)$ by applying the closed-loop complex dynamics to $\phi(i, x_0^c, \mathbf{u}^l)$ with the control determined by $\mathbf{u}^l$ to both sides,

$$x_i^l = \phi(i, x_0^c, \mathbf{u}^l) \tag{42}$$

$$f^c(x_i^l, \mathbf{u}_i^l) = f^c(\phi(i, x_0^c, \mathbf{u}^l), \mathbf{u}_i^l) \tag{43}$$

$$f^c(z_i^l) = \phi(i+1, x_0^c, \mathbf{u}^l) \tag{44}$$

Thus we need to show that $x^l_{i+1} = f^c(z^l_i)$ which would show that z^l_i would satisfy. We proceed by cases based on inclusion in S_k.

Case 1. $i \notin S_k, i+1 \notin S_k$. This case corresponds to a portion of the trajectory entirely in the complex space. By the definition of the adaptive system dynamics in (35),

$$x^l_{i+1} = x^a_{i+1} = f^c(z^a_i) = f^c(z^l_i) \tag{45}$$

Case 2. $i \notin S_k, i+1 \in S_k$. This case corresponds to a portion of the trajectory which decreases in complexity. By the definition of the adaptive system dynamics in (35) and the construction of S_k in (41),

$$x^l_{i+1} = \psi^\dagger_x(x^a_{i+1}) = \psi^\dagger_x \circ \psi \circ f^c(z^a_i) = f^c(z^a_i) = f^c(z^l_i) \tag{46}$$

Case 3. $i \in S_k, i+1 \notin S_k$. This case corresponds to a portion of the trajectory which increases in complexity. By the definition of the adaptive system dynamics in (35) and the construction of S_k in (41),

$$x^l_{i+1} = x^a_{i+1} = f^c \circ \psi^\dagger(z^a_i) = f^c(z^l_i) \tag{47}$$

Case 4. $i \in S_k, i+1 \in S_k$. This case corresponds to a portion of the trajectory entirely in the simple space. By the definition of the adaptive system dynamics in (35) and the construction of S_k in (41),

$$x^l_{i+1} = \psi^\dagger_x(x^a_{k+1}) = \psi^\dagger_x \circ f^s(z^a_i) = \psi^\dagger_x \circ f^s \circ \psi(z^l_i) = f^c(z^l_i) \tag{48}$$

Thus the induction step holds, completing the proof. □

We must also show that state and control trajectories which satisfy the adaptive state and control constraints $\mathcal{Z}^a$ also satisfy the complex equivalent $\mathcal{Z}^c$ by nature of the admissibility of set S_k.

Proposition 2. *Suppose Assumption 3 is satisfied. If $z_i^a \in \mathcal{Z}^a$, then $z_i^l \in \mathcal{Z}^c$.*

Proof. Proceed by cases based on inclusion in S_k.

Case 1. $i \notin S_k$. In this case, $z_i^l = z_i^a$. By the definition of $\mathcal{Z}^a$ in (36), $z_i^l \in \mathcal{Z}^c$.

Case 2. $i \in S$. In this case, $z_i^l = \psi^\dagger(z_i^a)$. Since $S_k \in \mathcal{S}_a$, $z_i^l \in \mathcal{Z}^c$ by (41b).

□

Next we show that satisfying the initial and terminal state constraints in the adaptive system implies satisfaction of the same constraints in the complex space.

Proposition 3. *Suppose Assumption 3 is satisfied. If $x_0^a = x_k^c$ then $x_0^l = x_k^c$, and if $x_{N^a}^a \in \mathcal{X}_t^c$, then $x_{N^a}^l \in \mathcal{X}_t^c$.*

Proof. Since $i = 0 \notin S_k$, $x_0^l = x_0^a = x_k^c$. Since $i = N^a \notin S_k, x_{N^a}^l = x_{N^a}^a \in \mathcal{X}_t^c$. □

4.5.2 Adaptive Complexity Feasibility and Stability

With these propositions in place we can now prove that the OCP defined in (37) is recursively feasible for states in $\mathcal{X}_{N^a}$ and that this set is invariant in the complex system. This is done by following the form of the proofs in [Allgöwer and Zheng, 2012], which constructs a feasible solution (in the absence of modeling errors) to the OCP at the successor state by combining the current solution with the terminal set feedback policy $u_t(x^c)$. This approach is illustrated in Fig. 27. Define this control sequence as $\tilde{\mathbf{u}}^a$ and its lifted counterpart $\tilde{\mathbf{u}}^l$,

$$\tilde{\mathbf{u}}^a(x_k^c) = \left[u_1^{*a}, \ldots, u_{N^a-1}^{*a}, u_t(\hat{x})\right] \tag{49}$$

$$\tilde{\mathbf{u}}^l(x_k^c) = \psi_u^\dagger(\tilde{\mathbf{u}}^a(x_k^c)) \tag{50}$$

where $\hat{x} = \phi(N^a, x_k^c, \mathbf{u}^{*l})$ is the terminal state at time N^a resulting from initial state x_k^c and control $\mathbf{u}^{*l}(x_k^c)$ (under Proposition 1). We now show that this control satisfies the requirements for recursive feasibility.

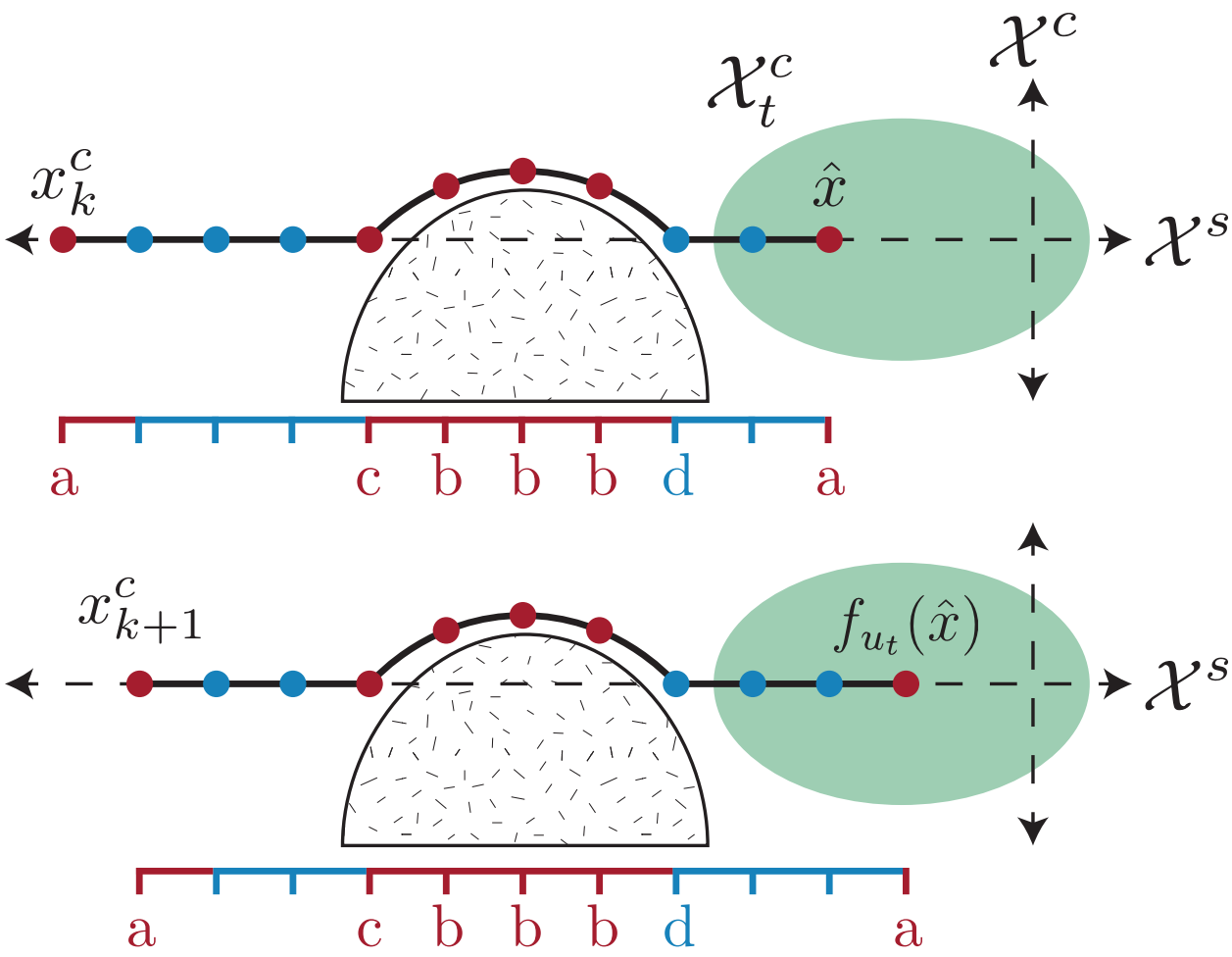


Figure 27: Adaptive complexity MPC retains recursive feasibility and admissibility by updating the simplicity set and solution at time $k+1$ with the corresponding terms from time k along with the new state and control determined from the terminal policy u_t which is applied with the terminal set $\mathcal{X}_t^c$. Letters at bottom indicate the condition of (41) that element satisfies or violates. The complex manifold $\mathcal{X}^c$ is the 2D space while $\mathcal{X}^s$ is the embedded 1D submanifold.

Proposition 4. *Suppose Assumptions 1 – 3 are satisfied. Let* $x_k^c \in \mathcal{X}_{N^a}$ *and let* $x_{k+1}^c \coloneqq f_{h^a}^c(x_k^c)$ *denote the successor state (under adaptive complexity model predictive control) to* x_k^c*. Then* $\tilde{\mathbf{u}}^a(x_k^c)$ *defined in (49) is feasible for* $\mathcal{P}_N^a(f_{h^a}^c(x_k^c))$ *and* $\mathcal{X}_{N^a}$ *is positively invariant (for the system* $x_{k+1}^c = f_{h^a}^c(x_k^c)$*).*

Proof. This proof follows standard methods for demonstrating recursive feasibility [Allgöwer and Zheng, 2012], see Appendix 4.9.1 for the complete proof. □

We must also show that the cost function decreases along any solution of $x_{k+1}^c = f_{h^a}^c(x_k^c)$ given

these previous assumptions as this is necessary for the stability proof:

Proposition 5. *Suppose Assumptions 1 – 3 are satisfied. Then*

$$V_N^{*a}(f_{h^a}^c(x_k^c)) - V_N^{*a}(x_k^c) \leq -L^c(x_k^c, h^a(x_k^c)) \tag{51}$$

Proof. See Appendix 4.9.1. □

We can now prove asymptotic stability of the origin of the closed loop system using standard Lyapunov-based methods. This supports Hypothesis 1 which states that adaptive complexity MPC yields provable stability properties reliant on template and anchor conditions.

Theorem 6. *Suppose Assumptions 1 – 3 are satisfied. Then there exists functions* $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{K}_\infty$ *which upper and lower bound the cost, i.e.,*

$$\alpha_1(|x_k^c|) \geq V_{N^a}^{*a}(x_k^c) \geq \alpha_2(|x_k^c|) \tag{52a}$$

$$V_{N^a}^{*a}(x_{k+1}^c) - V_{N^a}^{*a}(x_k^c) \leq -\alpha_3(|x_k^c|) \tag{52b}$$

and thus the origin of the system,

$$x_{k+1}^c = f_{h^a}^c(x_k^c) \tag{53}$$

is asymptotically stable with a region of attraction $\mathcal{X}_{N^a}$.

Proof. See Appendix 4.9.1. □

4.5.3 Recursive Admissibility of S_k

Theorem 6 shows that adaptive complexity MPC is stable under Assumption 3 which states that S_k is admissible. Next we prove that this holds for all time under Algorithm 1, again assuming no modeling errors. Robustness considerations remain an intriguing area for future investigation.

Lemma 7. *Let* $x_0^c \in \mathcal{X}_{N^a}$. *Then* $S_k \in \mathcal{S}_a \; \forall k \geq 0$ *under Algorithm 1.*

Proof. We proceed by induction. The base case $k = 0$ is met by the assumption that $S_0 \in \mathcal{S}_a$. For the induction step we must show that $S_k \in \mathcal{S}_a$ implies $S_{k+1} \in \mathcal{S}_a$. By Proposition 4, z_{k+1}^{*l} consists of each of the last $N^a - 1$ elements of z_k^{*l}, plus the new terminal state-control pair $(\hat{x}, u_t(\hat{x}))$. Since under Algorithm 1 elements of S_{k+1}^a are the time-shifted elements of S_k^a which satisfy the admissibility conditions (41), these conditions are satisfied for all $i \in S_{k+1}^a$. Since by the definition of $S_{k+1} = S_{k+1}^a \cap S^f$ where $i = N^a \notin S^f$, the new terminal state-action pair is always in the complex space, and hence (41) are satisfied for all $i \in S_{k+1}$ and thus $S_{k+1} \in \mathcal{S}_a$. □

Note that the terminal state-control pair may not meet the reduction conditions in (41), meaning that it must remain in the complex space. This is handled by assuming the last finite element in the horizon is complex, and checking (41) after solving the OCP to determine if the new index can be allowed into S_k or must remain complex.

4.5.4 Basin of Attraction Comparison

We have shown that both the original MPC formulation for the complex system in (29) and the adaptive complexity MPC system in (40) are asymptotically stable about the origin with basins of attraction $\mathcal{X}_{N^c}$ and $\mathcal{X}_{N^a}$, respectively. We now show that the size of their basins of attraction is dependent on the horizon lengths N^c and N^a.

Lemma 8. *If $N^c \leq N^a$ then $\mathcal{X}_{N^c} \subseteq \mathcal{X}_{N^a}$, and if $N^c < N^a$ then $\mathcal{X}_{N^c} \subset \mathcal{X}_{N^a}$.*

Proof. Let $x_0^c \in \mathcal{X}_{N^c}$, thus the OCP $\mathcal{P}_{N^c}(x_0^c)$ has a solution $\mathbf{u}^*$ satisfying the constraints in (27). Let S_0 be the initial admissibility set of this solution determined by evaluating the conditions in (41). Construct an adaptive control trajectory $\mathbf{u}^a$,

$$u_i^a = \begin{cases} u_i^* & i \notin S_0 \\ \psi_u(u_i^*) & i \in S_0 \end{cases} \tag{54}$$

for all $i = 0, 1, \ldots, N^c - 1$, which implies $\mathbf{u}^l = \mathbf{u}^*$. By Propositions 1 – 3 the constraints on the OCPs (27) and (37) are equivalent, and thus $\mathbf{u}^a$ is a valid solution of (37) which implies that

$x_0^c \in \mathcal{X}_{N^a}$ and thus $\mathcal{X}_{N^c} \subseteq \mathcal{X}_{N^a}$.

To show that $N^c < N^a \rightarrow \mathcal{X}_{N^c} \subset \mathcal{X}_{N^a}$, consider a point $x_0^c \notin \mathcal{X}_{N^c}$ and some feasible control $u_0^c \in \mathcal{U}$ such that $z_0^c \coloneqq (x_0^c, u_0^c) \in \mathcal{Z}^c$ and $x_1^c = f^c(z_0^c) \in \mathcal{X}_{N^c}$. Let u_1^* be the control sequence yielded by solving the OCP $\mathcal{P}_{N^c}(x_1^c)$. Let the control sequence $\mathbf{u}_0^a = [u_0^c, \mathbf{u}_1^a]$, and let $N^a = N^c + 1$. Since the state $x_1^c \in \mathcal{X}_{N^a}$, by Propositions 1 – 3 the state, control and terminal constraints of (37) are satisfied for $z_i^a \ \forall i = 1, 2, \ldots, N^c$, and since $z_0^c \in \mathcal{Z}^c$, the state and control constraints are satisfied for $i = 0$. Thus all the constraints of (37) are satisfied, and therefore $x_0^c \in \mathcal{X}_{N^a}$. The property $\mathcal{X}_{N^c} \subset \mathcal{X}_{N^a}$ follows. □

The property that longer horizon lengths yield larger basins of attraction is a known result of MPC theory – as horizon lengths go to infinity, MPC converges to infinite-horizon optimal control [Allgöwer and Zheng, 2012]. Horizon lengths are generally limited by computational effort, so if simplifying the problem allows for longer horizons under equal computational capabilities without violating system constraints, the resulting controller has a larger basin of attraction. We note this theoretical result, although in the following experiments we leave horizon length fixed and allow solve time variations rather than the reverse as this yields a more consistent representation of computational constraints. Also note that since membership in the basin of attraction implies that a solution to the OCP exists, Lemma 8 implies that adaptive complexity maintains the completeness properties of the original MPC formulation in the complex system (assuming a suitable algorithm which can solve the OCP in (37)).

Additionally, it should be noted that although feasible solutions of the adaptive OCP are also feasible solutions of original complex OCP, they may not be optimal with respect to the original OCP cost function. Reducing the system at a particular time is in essence constraining it to the simple manifold at that time. The conditions stated above require that doing so be feasible and yield a decreasing cost, yet it is possible that doing so could yield a trajectory of higher cost across the whole trajectory than if this constraint were lifted. As such this approach may result in a slight sacrifice in cost optimality in favor of a simpler problem.

4.6 Application to Legged Systems

To demonstrate the validity of the proposed adaptive complexity MPC in controlling dynamical systems and to provide examples for the quantities defined above, we apply the approach for a legged robot systems. This section defines the complex and simple models which are used for implementing the algorithm, as well as the mappings between them. The complex model includes states and constraints on both the body and feet of the robot (and thus also joint information through kinematics calculations) whereas the simple model only considers body motion.

4.6.1 Definition of Complex Legged System

The complex system represents a model of the robot that includes body and foot states as shown in Fig. 28. This formulation permits the calculation of joint kinematic data via known kinematics functions while maintaining some level of decoupling between body and leg dynamics. We define the states of the complex system $x^c \in \mathbb{R}^{12+6n}$ with n denoting the number of legs,

$$x^c = \begin{bmatrix} q_{\text{lin}} \\ q_{\text{ang}} \\ q_{\text{foot}} \\ \dot{q}_{\text{lin}} \\ \omega \\ \dot{q}_{\text{foot}} \end{bmatrix} \tag{55}$$

where q_{lin} defines the body linear position, q_{ang} defines the body orientation through a vector parameterization such as Euler angles, q_{foot} defines the foot positions, and ω is the angular velocity (with $\widehat{\omega}$ its skew-symmetric equivalent).

(a) Complex system (b) Simple system

Figure 28: The complex model includes body and foot states which together define joint states, while the simple model only consists of body states.

Define the control inputs $u^c \in \mathbb{R}^{6n}$,

$$u^c = \begin{bmatrix} u_{\text{body}} \\ u_{\text{foot}} \end{bmatrix} \tag{56}$$

where $u_{\text{body}} \in \mathbb{R}^{3n}$ are the desired ground reaction forces at each foot in the world frame coordinates, and $u_{\text{foot}} \in \mathbb{R}^{3n}$ denote the forces which accelerate the feet during swing, but do not act on the robot body. This parameterization decouples the effects of the body and leg dynamics, which amounts to a massless leg assumption for the body dynamics [**?**, Sec. 2.3] while still capturing kinematic constraints and second-order dynamics of the leg motion during swing. Note that these forces on the physical system correspond to the same actuators as any given leg is either in stance or swing – this separation is primarily to distinguish between control authority available in the simple and complex systems. Let $u_{\text{body},i} \in \mathbb{R}^3$ and $p_i \in \mathbb{R}^3$ be the ground reaction forces and foot positions respectively for stance leg i. Following the standard formulation for single rigid body dynamics models, e.g. as in [Chignoli and Wensing, 2020], define the continuous time dynamics

of this system $f^c(x^c, u^c)$,

$$f^c(x^c, u^c) = \begin{bmatrix} \dot{q}_{\text{lin}} \\ R(q_{\text{ang}})\omega \\ \dot{q}_{\text{foot}} \\ \frac{1}{m}\sum_i^n u_{\text{body},i} - g \\ W(q_{\text{lin}}, q_{\text{foot}}, \omega, u_{\text{body}}) \\ u_{\text{foot}} \end{bmatrix} \tag{57}$$

where $R(q_{\text{ang}}) \in SO(3)$ is the rotation matrix corresponding to orientation parameterization q_{ang}, m is the body mass, g is the gravity vector, and the shorthand function $W(\cdot)$ maps the state and control to the angular acceleration,

$$\begin{aligned} &W(q_{\text{lin}}, q_{\text{foot}}, \omega, u_{\text{body}}) = \\ &\qquad I^{-1}\Big(R^T \sum_j^n ((q_{\text{foot},j} - q_{\text{lin}}) \times u_{\text{body},j}) - \widehat{\omega} I \omega\Big) \end{aligned} \tag{58}$$

where I is the inertia matrix in the body frame. The discrete time formulation of the dynamics in (57) can be obtained with a suitable integration scheme such as forward Euler.

Next, we define the constraints of the complex system $\mathcal{Z}^c$. Kinematic constraints for legged systems are generally functions of joint limits rather than body or foot variables. As a result, we add the joint information to the optimization as slack variables and use them to define constraints. Let $\theta, \dot{\theta}, \tau \in \mathbb{R}^{n \cdot n_j}$ be the joint positions, velocities, and torques, respectively, where n_j is the

number of joints per leg. These slack variables are constrained,

$$FK(q, \theta) = q_{\text{foot}} \tag{59a}$$

$$J(q, \theta)[\dot{q}_{\text{lin}}^T, \omega^T, \dot{\theta}^T]^T = \dot{q}_{\text{foot}} \tag{59b}$$

$$\tau = -J_{\text{body}}(q, \theta)^T u_{\text{body}} \tag{59c}$$

where $FK(q, \theta)$ is the forward kinematics function of the system and $J(q, \theta)$ is the leg Jacobian which relates motion of the body and joints to foot motion in the world frame, and J_{body} are the columns of this matrix corresponding to joint motion.

With these variables in place we can now state the constraints in the complex system,

$$\theta_{\text{min}} \leq \theta \leq \theta_{\text{max}} \tag{60a}$$

$$\dot{\theta}_{\text{min}} \leq \dot{\theta} \leq \dot{\theta}_{\text{max}} \tag{60b}$$

$$\tau_{\text{min}} \leq \tau \leq \tau_{\text{max}} \tag{60c}$$

$$u_{\text{body,min}} \leq u_{\text{body}} \leq u_{\text{body,max}} \tag{60d}$$

$$Du_{\text{body}} = 0 \tag{60e}$$

$$u_{\text{body}} \in \mathcal{FC} \tag{60f}$$

$$-\tau_{\text{max}} \left(1 + \frac{\dot{\theta}}{\dot{\theta}_{\text{max}}}\right) \leq \tau \leq \tau_{\text{max}} \left(1 - \frac{\dot{\theta}}{\dot{\theta}_{\text{max}}}\right) \tag{60g}$$

$$h(q_{\text{foot}}) \geq 0 \tag{60h}$$

where $(\cdot)_{\text{min}}$ and $(\cdot)_{\text{max}}$ represent variable bounds, (60e) enforces a contact schedule with selection matrix D, (60f) enforces that the GRF at each foot lies within the non-adhesive friction cone $\mathcal{FC}$, (60g) enforces a linear motor model, and (60h) enforces non-penetration of the terrain via the ground clearance $h(q_{\text{foot}})$. Together the constraints in (60) define the set $\mathcal{Z}^c$.

The stage and terminal costs for the OCP of the complex system can then be defined,

$$L^c(x^c, u^c) = x^{cT} Q x^c + u^{cT} R u^c \tag{61}$$

$$V_f^c(x^c) = x^{cT} Q_t x^c \tag{62}$$

where Q, Q_t, and R are positive definite matrices.

4.6.2 Definition of Simple Legged System

The simple system represents the reduced-order model of the robot that uses only the body states and ignores the foot and joint states, as shown in Fig. 28. This is a commonly used model reduction technique in the legged locomotion literature. The state of the simple system is defined as $x^s \in \mathbb{R}^{12}$. The control inputs $u \in \mathbb{R}^{3n}$ of the simple system consists of only the GRFs from the complex system, such that $u^s = u_{\text{body}}$. The dynamics in the simple system are the components of the complex dynamics corresponding to the simple system states defined as $f^s(x^s, u^s)$. The constraints in the simple space are only constraint bounds on the input u_{body}, identical to equations (60d)–(60f). Note that the introduction of the torque constraints in (60c) and (60g) accurately capture actuation limits, and thus the heuristic GRF limits in (60d) can be selected optimistically in the simple case as the system will adaptively apply the more accurate constraints as needed to ensure feasibility.

Although the cost in the adaptive OCP is based on the cost function defined for the complex system, we define the stage and terminal costs of the simple system as L^s and V_t^s which are used in the following experiments for the non-adaptive configurations. Note that the cost for the simple system is structurally identical to the complex system with the difference in the states, controls, Q, R, and Q_t matrices representing the simple system variables instead.

4.6.3 Relations Between Complex and Simple Legged Systems

With these systems defined, we can now relate the two with the projections ψ_x, ψ_u and define our heuristic lifts $\psi_x^\dagger, \psi_u^\dagger$. The projections select components to retain within the simple system,

$$\psi_x(x^c) = \begin{bmatrix} q_{\text{lin}} \\ q_{\text{ang}} \\ \dot{q}_{\text{lin}} \\ \omega \end{bmatrix} = x^s \tag{63}$$

$$\psi_u(u^c) = [u_{\text{body}}] = u^s \tag{64}$$

In order to define the lifting functions, assume that we have a reference $\bar{x}_k^c, \bar{u}_k^c$ in the complex system space which is dynamically consistent, i.e. $\bar{x}_{k+1}^c = f^c(\bar{x}_k^c, \bar{u}_k^c)$. Although Section 4.5 assumes a time-invariant system (and thus a constant reference) for simplicity of the analysis, the results could be applied to time-varying systems and thus the tracking of a trajectory. We leave the formal extension to these systems as future work, but note that the stability of MPC for time-varying systems is well established [Rawlings et al., 2017]. Dropping the index k for clarity, this reference allows the definition of the lifting operator $\psi^\dagger$,

$$\psi_x^\dagger(x^s) = \begin{bmatrix} q_{\text{lin}} \\ q_{\text{ang}} \\ \bar{q}_{\text{foot}} \\ \dot{q}_{\text{lin}} \\ \omega \\ \dot{\bar{q}}_{\text{foot}} \end{bmatrix} \tag{65}$$

$$\psi_u^\dagger(u^s) = \begin{bmatrix} u_{\text{body}} \\ \bar{u}_{\text{foot}} \end{bmatrix} \tag{66}$$

Note that when tracking a reference, the state is typically mapped to tracking error such that stability of the origin corresponds to stability of the reference trajectory, i.e. $\hat{x}^c = x^c - \bar{x}^c$. We can now state the conditions for admissibility for this system, which require the that variables in the null space of ψ lie on the reference and are feasible.

Lemma 9. *For a given state-control pair z_i^l which lie on a trajectory of the system defined in* (57)*, a reduction at $i \in [1, \ldots, N-1]$ is admissible (Definition 1) if $q_{\text{foot}} = \bar{q}_{\text{foot}}$, $\dot{q}_{\text{foot}} = \dot{\bar{q}}_{\text{foot}}$, $u_{\text{foot}} = \bar{u}_{\text{foot}}$, and z_i^l satisfies the constraints in* (60)*.*

Proof. See Appendix 4.9.2. □

4.7 Experimental Evaluation

This section presents experiments deploying adaptive complexity MPC on a simulated quadrupedal robot to quantify its performance and benchmark against other formulations of MPC. In particular we compare against three other model configurations – "Simple" and "Complex" respectively employ only the simple and complex model dynamics and constraints, and "Mixed" employs the complex model for the first one-quarter of the horizon and the simple model for remainder, similar to [Li et al., 2021]. The "Adaptive" configuration follows Algorithm 1 with $S^f = \{2, ...N^a - 2\}$ so that the first and last finite elements are always complex. This meets the terminal state admissibility condition in (41a) and also ensures that new elements entering the horizon will be complex to meet Assumption 3 in lieu of a hard-to-find terminal controller $h_t(x)$ that meets the conditions in Assumption 2.

In each experiment, the robot is provided a reference trajectory which defines the required task over the given environment. We use three environments to evaluate the algorithm performance in the presence of varying constraints. The "Acceleration" environment requires the robot to rapidly accelerate and decelerate over 7.5 body lengths (3 m) of flat terrain to measure the ability of the configuration to stabilize the system during agile motions. The "Step" environment consists of a one-half leg length (20 cm) step which requires navigating state constraints such as joint limits

to traverse. The "Gap" environment consists of a body-length (40 cm) gap which the robot must leap across, testing the controller's ability to handle these kinematic constraints in addition to input constraints such as actuator limits and friction.

We quantify performance in the Acceleration environment by measuring the average completion time and resulting top speed achievable with a 100% success rate over ten trials. Success is defined as reaching the goal within a one-half body length (20 cm) and zero velocity without the body contacting the ground. Performance in the Step and Gap environments is measured via the success rate over ten trials, as well as average solve time and norm of the total control input (sum of ground reaction forces) averaged over the successful trials. For the Step and Gap environments, the robot is initialized to a random position within one-half body length in the transverse plane from a nominal position.

In each environment a fixed reference trajectory for the body is provided from the global planner described in [Norby and Johnson, 2020], and the reference foot trajectories are chosen online before each MPC iteration with a Raibert-like heuristic [Raibert, 1986] and a threshold on traversability, as described in [Norby et al., 2022b]. We fix the prediction horizon at two gait cycles ($N^c = N^a = 24$) with a timestep of $\Delta t = 0.03$ s and measure the relative solve times as discussed in Sec. 4.5.4, although future work could implement an adaptive horizon approach to keep solve time fixed.

Once the reference information has been obtained, we construct the NLP with the appropriate complexity structure and solve it with IPOPT [Wächter and Biegler, 2006]. We configure IPOPT to enable warm start initialization and provide it the primal and dual variables from the prior solve (appropriately shifted) for rapid convergence. Once the problem is solved, the MPC control output is then mapped from ground reaction forces to joint torques via the Jacobian-transpose method, and the resulting swing foot trajectories are tracked with PD control. See [Norby et al., 2022b] for more details on the implementation of the low-level controller. All experiments were performed using Gazebo 9 with the ODE physics engine, and all processes were executed on a machine running Ubuntu 18.04 with an Intel Core i7-12700K CPU at 4.9 GHz and with 64 GB of RAM.

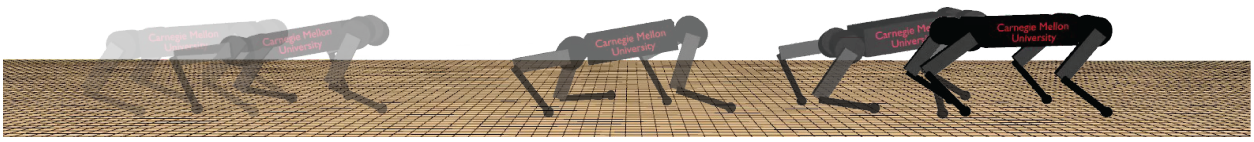


Figure 29: The Acceleration environment requires the robot to rapidly move forwards 3 m and come to a rest. Snapshots are equally distributed in time, with increasing opacity corresponding to progression forwards in time.

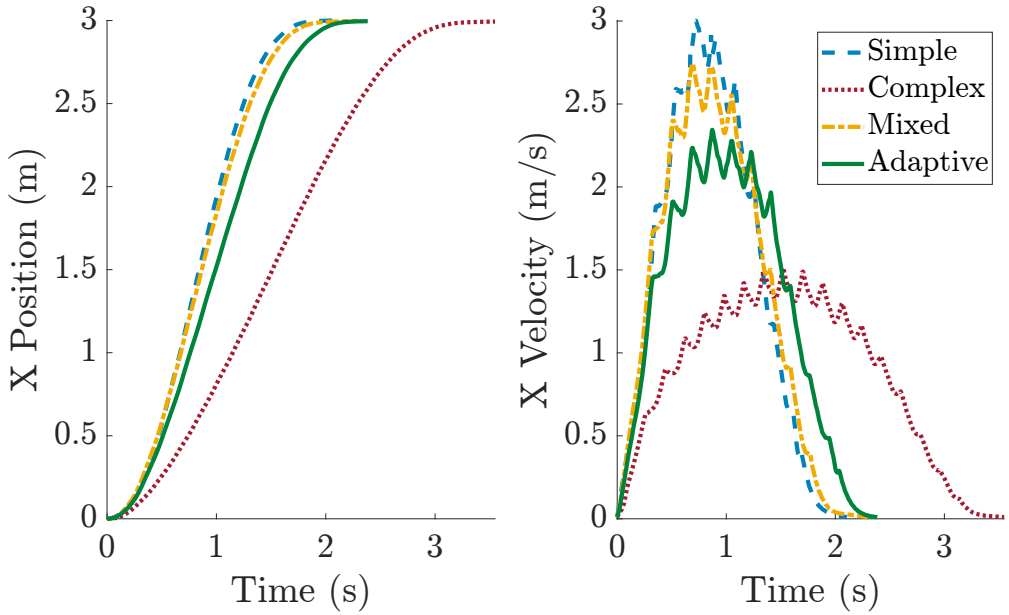


Figure 30: The position and velocity trajectories for the Acceleration experiment show that the additional computation required by the Complex configuration significantly reduces its performance. Each curve corresponds to one trial at the maximum feasible commanded acceleration.

The Acceleration environment is simulated in real-time to measure the effect of solve time on stability. The simulations were slowed down by a factor of 2x for the Step environment and 5x for the Gap environment (with a maximum solve time of $4\Delta t = 0.12$ s) since resolving the constraints in these tasks are still computationally intensive.

4.7.1 Acceleration Environment

A series of snapshots of the Adaptive configuration performing the Acceleration task are shown in Fig. 29. Since the peak acceleration of the system occurs at the beginning and end of the trajectory, rapidly converging on a feasible solution to the OCP is essential. Crucially, performing this task does not require exact knowledge of the joint constraints and thus computational efficiency is key.

Table 3: Experimental data for the Acceleration environment

Config	% Horizon Simplified	Completion Time (s)	Max Velocity (m/s)
Simple	100	2.0	3.0
Complex	0	3.5	1.5
Mixed	75	2.2	2.7
Adaptive	86	2.4	2.3

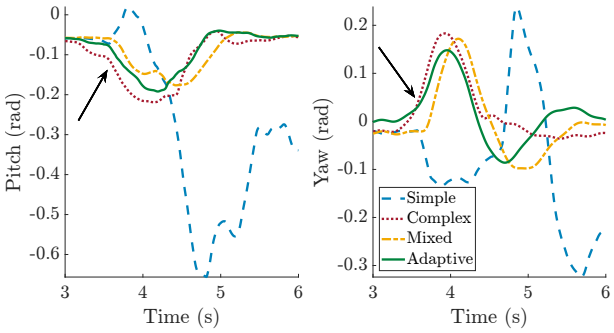
Results for each configuration are shown in Table 3 with state trajectories of candidate trials in Fig. 30. The Simple configuration exhibits the best performance with a 100% increase in top speed over Complex and a 43% reduction in completion time. Mixed performs next best at an 80% increase in top speed and 37% reduction in completion time, followed by Adaptive at a 50% increase in top speed and 31% reduction in completion time. These reflect the relative complexity of each configuration – since the Complex system must reason about extraneous constraints over the entire horizon, it takes longer to solve the problem and is thus less capable of stabilizing high-acceleration behaviors. The Simple configuration conversely excels since it is solving a reduced problem. The Mixed and Adaptive systems consist mostly of simple finite elements and thus retain this benefit, although the Adaptive configuration performance is slightly degraded since more complex elements are added at the beginning of the behavior during the period of high acceleration. These results support Hypothesis 3 which states that the reducing the model yields improved locomotion performance through more efficient computation.

4.7.2 Step Environment

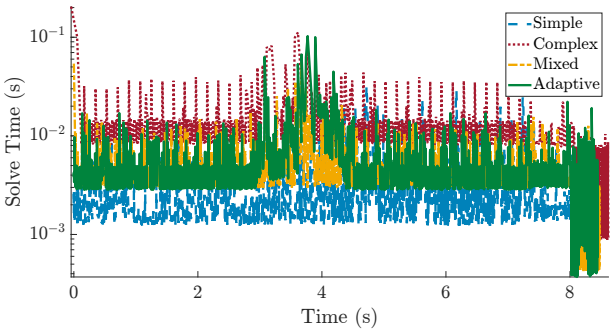
A series of snapshots of the Adaptive configuration navigating the Step environment are shown in the top row of Fig. 31a. The key constraints which must be resolved are the joint limits of the robot and the height of the toe, as the system must ensure the toe clears the step while also ensuring the rear legs can still reach the terrain for support.

Results from each MPC configuration are shown in Fig. 31 and quantitatively summarized in Table 31f. The state trajectories are shown in Fig. 31b. The Simple, Mixed, and Adaptive configu-

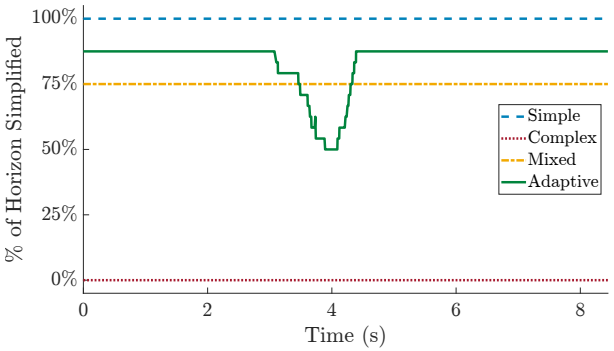
(a) The Step environment requires navigating kinematic constraints. Snapshots are shown of the trajectories under the Adaptive configuration. Non-steppable regions are indicated with darker shading.



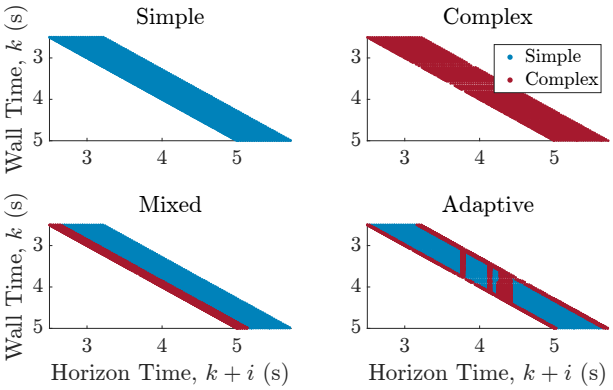
(b) Step environment state trajectories. Complex (red) and Adaptive (green) show changes to pitch and yaw (indicated by the arrows) before the Simple (blue) and Mixed (gold) configurations.



(c) Step environment solve times. The increase four seconds into the behavior corresponds to navigating the step.



(d) Step environment horizon simplification percentage. The Adaptive configuration is able to simplify the problem for most of the behavior, and quickly recover these simplifications once the difficult behavior is resolved.



(e) Step environment prediction horizons. Horizons at each time are indicated by horizontal slices of finite elements (dots), where dot color indicates model complexity. The vertical bands of increased complexity correspond to joint singularities.

Config	Success Rate	Mean Solve Time (ms)	Slow Solve Rate (%)	Mean Control (N)
Simple	10/10	2.2	0.039	43
Complex	9/10	11	16	16
Mixed	7/10	4.3	3.9	22
Adaptive	10/10	4.3	2.6	17

(f) Experimental data for the Step environment.

Figure 31: Data for Step environment. The Adaptive configuration is able to leverage admissible reductions for the majority of the behavior while retaining the ability to react quickly to the kinematic constraints required to navigate the step.

rations are able to complete the task and reach the goal, while the Complex configuration fails due to excessive solve times. However, the lack of constraint information in the Simple configuration and the myopia of the Mixed configuration result in large control actions when crossing the step which nearly destabilize the system. Meanwhile, the Complex and Adaptive configurations are able to see the step sooner and react by increasing the walking height and rotating the body to more safely navigate the step (arrows in Fig. 31b), although only the Adaptive framework does so while respecting solve time constraints.

The computational effort of each configuration is shown in Fig. 31c. Unsurprisingly the Simple configuration is consistently the fastest since its model is the most sparse and it is unaware of the nonlinear joint kinematic constraints, while the Complex configuration consistently takes the longest, especially to find an initial solution and also once it sees the step. Both the Mixed and Adaptive configurations have intermediate nominal solve times, but differ when the step approaches. The Adaptive configuration immediately takes much longer to solve the problem as it needs to reason about this new information, but once a valid solution is found it settles back to its nominal solve time as more complex elements are converted back to simple ones. Meanwhile the Mixed formulation only increases in solve time when the step is within its shorter window of complex elements, and planning the large control forces required to navigate the step on such short notice causes a large and sustained increase in solve time.

The degree of horizon simplification is shown in Fig. 31d. While the fixed-complexity configurations remain uniform for the entire task, the Adaptive configuration clearly leverages additional complexity when encountering the step. However, even in the worst case around half of the horizon remains simplified, the effects of which are seen in the lower solve times compared to the Complex configuration. These horizons are visually shown in Fig. 31e, which shows the prediction horizon at each time with complex and simple elements distinguished by different colors. The Adaptive configuration clearly changes around $k = 2.5$s as the step comes into view. As elements in the terminal region require leaving the simple manifold, adaptive complexity MPC fills in new elements with additional complexity. When the terminal region returns to the simple manifold, the

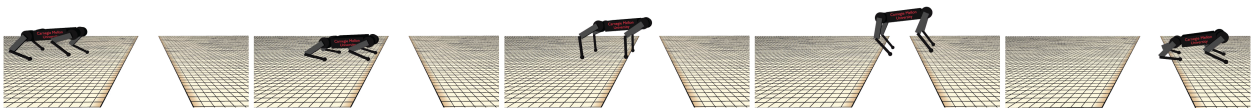
algorithm recognizes this and allows simple elements back into the horizon. Together, these results support Hypothesis 2 which states that reductions are frequently admissible for candidate terrains, and Hypothesis 3 which states that capturing the complex dynamics and constraints expands the range of executable tasks.

4.7.3 Gap Environment

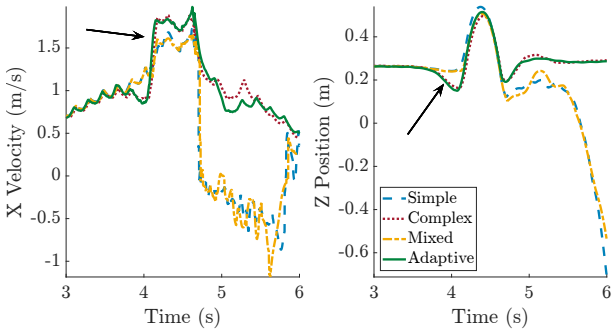
A series of snapshots of the Adaptive configuration navigating the Gap environment are shown in Fig. 32a. Due to the state-dependent actuator limits which reduces peak torque as joint velocity increases, the system must accelerate early to ensure enough velocity to land safely on the other side. Joint kinematics also limit how far forward the legs can reach to prepare for landing, so the system must be aware of these constraints before takeoff to ensure sufficient controllability.

Results for each configuration in the Gap environment are shown in Figs. 32 and quantitatively summarized in Table 32f. Only the Adaptive configuration has both the efficiency and model fidelity required to solve this task reliably. The Complex configuration fails in the majority of the trials due to its excessive computational effort, while the Simple and Mixed configurations never succeed because they lack the requisite constraint knowledge and thus do not leap far enough. Like in the Step environment, the configurations which are able to recognize constraints near the end of the horizon (Adaptive and Complex) do so immediately by lowering towards the ground to obtain a longer leaping stroke and accelerating forwards to ensure enough velocity to reach the other side of the gap, as shown in Fig. 32b.

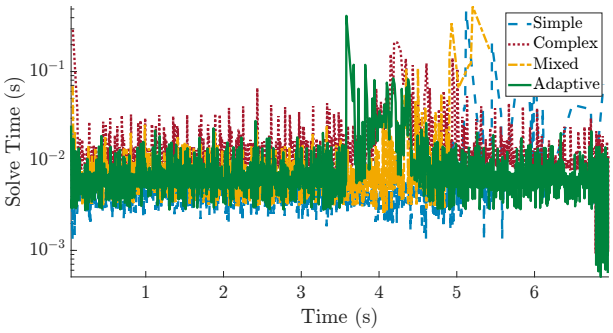
The solve times for the Gap environment shown in Fig. 32c demonstrate similar trends as during the Step environment. Both Simple and Mixed formulations maintain their performance until they fail to cross the gap, which quickly causes failed solves. Both Complex and Adaptive configurations have periods of longer solves to plan the leaping and landing phase, but the Adaptive formation is able to recover faster solve times sooner due to its ability to convert complex element to simple ones near the end of the leap. This is further supported by the data in Table 32f which shows a threefold reduction in Adaptive solve times exceeding one timestep compared to the Com-



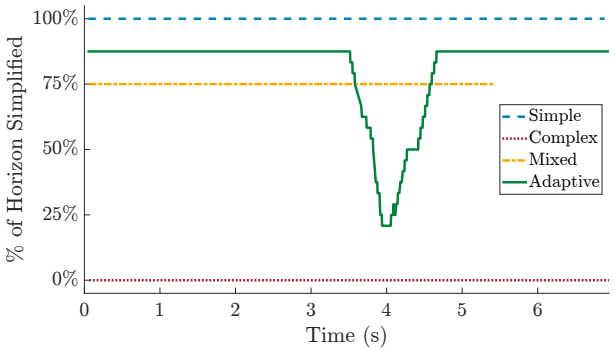
(a) The Gap environment requires navigating both kinematic and dynamic constraints. Snapshots are shown of the trajectories under the Adaptive configuration.



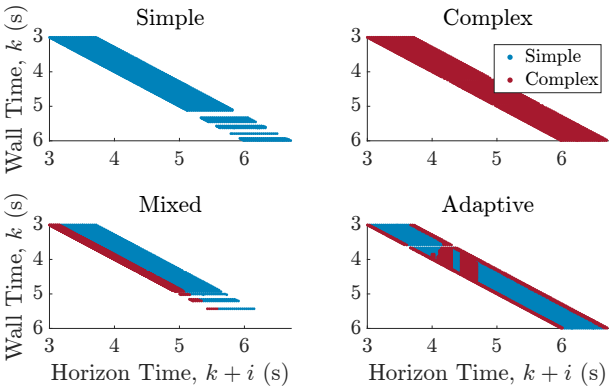
(b) Gap environment state trajectories. Complex (red) and Adaptive (green) show changes to horizontal velocity and vertical position (indicated by the arrows) before Simple (blue) and Mixed (gold).



(c) Gap environment solve times. The increase four seconds into the behavior corresponds to navigating the gap. The sustained increases for Simple and Mixed correspond to failed solves after a short landing.



(d) Gap environment horizon simplification percentage. Similarly to the Step environment, the Adaptive configuration is able to simplify most of the horizon, with the most complexity occurring when both takeoff and touchdown are within the horizon.



(e) Gap environment prediction horizons. Horizons at each time are indicated by horizontal slices of finite elements (dots), where dot color indicates model complexity. The vertical bands of increased complexity correspond to takeoff and touchdown.

Config	Success Rate	Mean Solve Time (ms)	Slow Solve Rate (%)	Mean Control (N)
Simple	0/10	–	–	–
Complex	9/10	13	13	33
Mixed	0/10	–	–	–
Adaptive	10/10	7.5	4.3	31

(f) Experimental data for the Gap environment.

Figure 32: Data for Gap environment. The Adaptive and Complex configurations are able to reason about constraints at the end of the horizon, allowing them to alter the leap to increase forward velocity and successfully land.

plex configuration. The mechanism for this reduction is further illustrated in the simplification percentages shown in Fig. 32d and the prediction horizons shown in Fig. 32e. Even in the worst-case portion of the behavior, the Adaptive configuration retains 25% simplification of the horizon, and once more feasible elements begin entering the horizon the Adaptive configuration can take advantage of the reduced complexity to improve solve times. These results support Hypotheses 2 and 3.

4.8 Conclusion

This work presents a formulation of adaptive complexity MPC which actively identifies regions where dynamics and constraints can be simplified without compromising the feasibility or stability of the original system. Analysis of the proposed approach demonstrates that under key conditions these simplifications do not compromise the stability properties of the original system, and can enable new behaviors by acting quickly to perform agile motions or looking further into the future to execute behaviors. These advantages are demonstrated on a simulated quadrupedal robot performing agile behaviors with challenging environmental constraints, and in particular expanding the leaping capability through receding horizon execution with knowledge of joint constraints.

While the MPC formulation presented here was primarily evaluated in locomotion applications, future work could investigate its applicability to other domains that employ hierarchical structures, such as manipulation. For example, often in manipulation settings the internal joints of the manipulator are neglected and planning is primarily conducted in the space of object motions and forces. Adaptive complexity MPC would enable an efficient handling of manipulator kinematics only when necessary so that the system can respect these constraints while largely retaining the benefits of improved efficiency, including faster reactions to unexpected object motion or longer planning horizons.

A primary limitation of adaptive complexity MPC is its reliance on the formulation of the additional constraints and dynamics of the complex system. Introducing additional numerical complexity such as non-convexity can make solving the OCP more susceptible to local minima or poor

convergence rates, which are then transferred to the adaptive configuration. This is most notable in the Step environment, in which the Simple configuration demonstrated remarkable ability to complete the task without constraint knowledge due to its numerical robustness. Ongoing work into well-conditioned OCP formulations or methods which identify which constraints are most necessary would benefit the approaches discussed here.

Another current drawback of adaptive complexity is robustness to unexpected errors in the simplicity set caused by model mismatch or disturbances. In particular, introducing additional complexity in the interior of the horizon can degrade the initialization of the OCP – this could be alleviated by recent approaches which warm-start the OCP with experiential data [Mansard et al., 2018], or possibly avoided by applying robust MPC techniques [Bemporad and Morari, 1999]. In addition, many hierarchical systems leverage reduced-order models to generate reference trajectories entirely in the simple system, making infeasible references highly relevant. Investigations into adapting model complexity to handle infeasible references such as [Batkovic et al., 2021] would be crucial to expand this work to a broader class of systems.

4.9 Appendix

4.9.1 Proofs Required for Adaptive Complexity MPC Stability

The proof of Proposition 4 shows recursive feasibility by finding a feasible solution to the adaptive OCP at the successor state.

Proof (Proposition 4). Since $x_k^c \in \mathcal{X}_{N^a}$, there exists a solution $\mathbf{u}^{*a}(x_k^c)$ to $\mathcal{P}_N^a(x_k^c)$. Because $\mathbf{u}^{*a}(x_k^c)$ is a solution of (37), by Proposition 3 the corresponding predicted terminal state $\hat{x} \in \mathcal{X}_t^c$. Let the successor control sequence $\tilde{\mathbf{u}}^a(x_k^c)$ be defined by (49). We claim this sequence is feasible for the OCP $\mathcal{P}_N^a(x_{k+1}^c)$ solved at successor state $x_{k+1}^c \coloneqq f_{h^a}^c(x^c)$.

Firstly, the controls $u_1^{*l}, \ldots, u_{N^a-1}^{*l}$ which are elements of $\mathbf{u}^{*l}(x_k^c)$ which was a solution to (37), all lie in $\mathcal{Z}^c$ by Proposition 2. It follows from Assumption 2 (since $\hat{x} \in \mathcal{X}_t^c$) that $u_t(\hat{x}) \in \mathcal{Z}^c$, and thus every element of $\tilde{\mathbf{u}}^l(x^c)$ satisfies the control constraint of (37).

Next we consider the state constraint. By Proposition 1 the state sequence resulting from initial state x_k^c and control sequence $\tilde{\mathbf{u}}^l(x_k^c)$ is $\tilde{\mathbf{x}}^c \coloneqq [\tilde{x}_0^c, \tilde{x}_1^c, \ldots, \tilde{x}_{N^a}^c]$ where,

$$\tilde{x}_j^c = x_{j+1}^{*l}, \quad j = 0, \ldots, N^a - 1 \tag{67}$$

$$\tilde{x}_{N^a}^c = f_{u_t}^c(\hat{x}) \tag{68}$$

and $\tilde{x}_0^c = x_1^{*l} = f^c(x^c, u_{x^c}^{*l}(0)) = f_{h^a}^c(x^c)$. By Proposition 2, the states $x_1^{*l}, \ldots, \hat{x}$ satisfy the state constraint. Since $\hat{x} \in \mathcal{X}_t^c$, Assumption 2 implies that $f_{u_t}^c(\hat{x}) \in \mathcal{X}_t^c \subset \mathcal{X}_{N^c}$, so that every element of the state sequence $\tilde{\mathbf{x}}^c = [\tilde{x}_0^c, \tilde{x}_1^c, \ldots, \tilde{x}_{N^a}^c]$ satisfies the state constraint, and the new terminal state $\tilde{x}_{N^a}^c = f_{u_t}^c(\hat{x})$ satisfies the stability constraint. Hence $\tilde{\mathbf{u}}^l(x^c)$ is feasible for $\mathcal{P}_N^a(f_{h^a}^c(x_k^c))$ and $f_{h^a}^c(x_k^c) \in \mathcal{X}_{N^a}$. □

The proof of Proposition 5 shows that the OCP cost function decreases along the closed loop system by leveraging shared terms in the solutions along with the observation that the cost of a state in the simple system is equal to the cost of that state lifted into the complex system.

Proof (Proposition 5). The sequence pairs $(\mathbf{u}^{*a}(x_k^c), \tilde{\mathbf{u}}^a(x_k^c))$ and $(\mathbf{x}^{*a}, \tilde{\mathbf{x}}^a)$ have common elements and thus the cost sequence can be simplified,

$$\begin{aligned}
V_{N^a}^{*a}(f_{h^a}^c(x_k^c)) - & V_{N^a}^{*a}(x_k^c) \\
&\leq V_{N^a}^a(x_{k+1}^c, \tilde{\mathbf{u}}^a(x^c)) - V_{N^a}^a(x^c, \mathbf{u}^{*a}(x^c)) \\
&= \big(L^c(\hat{x}, u_t(\hat{x})) + V_t(f_{u_t}(\hat{x}))\big) \\
&\quad - \big(L^c(x^c, h^a(x^c)) + V_t(\hat{x})\big) \\
&\quad + \big(L^c(x_{k+1}^c, u_1^{*a}) - L^a(x_{k+1}^c, u_1^{*a})\big) \\
&= - L^c(x^c, h^a(x^c)) + L^c(\hat{x}, u_t(\hat{x})) \\
&\quad + V_t(f_{u_t}(\hat{x})) - V_t(\hat{x})
\end{aligned} \tag{69}$$

where the property $L^c(x_{k+1}^c, u_1^{*a}) - L^a(x_{k+1}^c, u_1^{*a}) = 0$ comes from the definition of the adaptive

cost in (38) and the observation that $\psi^\dagger$ maps to the origin for variables in the null space of ψ. Since $\hat{x} \in \mathcal{X}_t^c$, Assumption 2 implies,

$$V_t(f_{u_c}(\hat{x})) - V_t(\hat{x}) \leq -L^c(\hat{x}, u_t(\hat{x})) \tag{70}$$

Hence (51) is satisfied for every $x_k^c \in \mathcal{X}_{N^a}$. □

The proof of Theorem 6 follows from the prior propositions:

Proof (Theorem 6). The inequalities in (52a) follow from the structure of the value function defined in Assumption 2. The descent inequality in (52b) is given by the structure of the value function along with Proposition 5. Asymptotic stability of the origin with region of attraction $\mathcal{X}_{N^a}$ follows from standard Lyapunov theory [Rawlings et al., 2017, Appendix B]. □

4.9.2 Proof of admissibility conditions for legged system

Proof (Lemma 9). The conditions under which the legged system described in Section 4.6 can be admissibly simplified rely on the feasibility of the reference trajectory. Lemma 9 states that an index within a lifted trajectory of the legged system can be admissibly reduced if the lifted components of the state-control pair $q_{\text{foot}}, \dot{q}_{\text{foot}}$, and u_{foot} all lie on the trajectory and satisfy the constraints in (60). We proceed by each condition for admissibility defined in (41), noting that (41a) is trivially satisfied by the assumptions of the lemma.

The condition in (41b) requires that constraints would be satisfied if the system were reduced, i.e. $\psi^\dagger \circ \psi(z_i^l) \in \mathcal{Z}^c$. By the conditions given on the values of z_i^l in the null space of ψ and the definition of $\psi^\dagger$, it follows that $\psi^\dagger \circ \psi(z_i^l) = z_i^l$. Since $z_i^l \in \mathcal{Z}^c$, it follows that $\psi^\dagger \circ \psi(z_i^l) \in \mathcal{Z}^c$.

The condition in (41c) requires that the complex system is exactly anchored by the simple system at that index in the trajectory, i.e. $\psi_x^\dagger \circ f^s \circ \psi(z_i^l) = f^c(z_i^l)$. We show this by directly

applying the dynamics and mappings in Section 4.6, dropping the index i for simplicity,

$$\begin{aligned}
\psi_x^\dagger \circ f^s \circ \psi(z^l) &= \psi_x^\dagger \circ \begin{bmatrix} \dot{q}_{\text{lin}} \\ R(q_{\text{ang}})\omega \\ \frac{1}{m}\sum_j^n u_{\text{body},j} - g \\ W(q_{\text{lin}}, \bar{q}_{\text{foot}}, \omega, u_{\text{body}}) \end{bmatrix} \\
&= \begin{bmatrix} \dot{q}_{\text{lin}} \\ R(q_{\text{ang}})\omega \\ \dot{\bar{q}}_{\text{foot}} \\ \frac{1}{m}\sum_j^n u_{\text{body},j} - g \\ W(q_{\text{lin}}, \bar{q}_{\text{foot}}, \omega, u_{\text{body}}) \\ \bar{u}_{\text{foot}} \end{bmatrix} \\
&= f^c(z^l)
\end{aligned}$$

Lastly, the condition in (41d) requires that the dynamics at the prior state lead to the manifold, i.e. $\psi_x^\dagger \circ \psi \circ f^c(z_{i-1}^l) = f^c(z_{i-1}^l)$. Since the trajectory z^l is valid for the system in (57), $x_i^l = f^c(z_{i-1}^l)$. Additionally, since $\psi^\dagger \circ \psi(z_i^l) = z_i^l$, it follows that $\psi_x^\dagger \circ \psi(z_i^l) = x_i^l$. Thus $\psi_x^\dagger \circ \psi \circ f^c(z_{i-1}^l) = f^c(z_{i-1}^l)$. □

5 Aerodynamic Tail Design

While motion planning and control algorithms are important for completing successful legged locomotion tasks, deviations from those plans are inevitable. Whether caused by errors in modeling, inaccurate sensor data, unpredicted changes in the environment, or a literal disturbance such as unexpected collision with the environment, a legged robot will inevitably need to react to unplanned phenomena. If these unexpected disturbances occur when legs are on the ground, a reactive controller like that discussed in Section 3 can partially mitigate errors. However, these disturbances could occur when few or no feet are on the ground, making rapid momentum regulation difficult. This underactuation can be fought by introducing additional actuation that does not rely on contact with the environment, such as that provided by a tail. This section explores the effectiveness of a novel tail design that harnesses aerodynamic drag to regulate angular momentum, rendering them lighter and more practical than standard inertial tails [Norby et al., 2021].

5.1 Introduction

Terrestrial animals often use tails to help a wide variety of behaviors that are traditionally challenging for robots. Agama lizards use their long, heavy tails to reorient in mid-air after unexpected foot slip, a failure mode which is often fatal for legged robots [Atkeson et al., 2018; Libby et al., 2012]. Kangaroos use their tails for stability when hopping and even support themselves with their tails while walking [Alexander, 1975; Donelan et al., 2014]. Although humans do not have tails, we swing our arms to the same effect while walking and running, increasing both lateral balance and energy efficiency [Arellano and Kram, 2011].

While these animals exhibit impressive behaviors, none approach cheetahs in terms of speed and agility. A cheetah chasing prey can reach a top speed of 29 m/s and accelerate, decelerate, or turn at rates almost double that of horses [Wilson et al., 2013]. These dynamic maneuvers require very precise regulation of angular momentum to maintain balance, avoid foot slip, and accommodate the large reaction forces and moments generated by the legs. Cheetahs have been

observed to flick their tails to aid this angular momentum regulation, particularly while decelerating and turning, as shown in Fig. 33 and in [Patel and Braae, 2013, 2014].

Inspired by the ability of these and other tailed animals, researchers have replicated similar tasks on robotic systems. The first known application of a tail in robotics was the Uniroo [Zeglin, 1991], a running robot that employed a tail to stabilize pitch. This example has since inspired orientation stabilization tails in many legged robots [Briggs et al., 2012; Haldane et al., 2017; Liu et al., 2014]. Platforms such as the Berkeley Tailbot, the MSU Tailbot, and Penn RHex have employed tails to successfully perform aerial self-righting maneuvers similar to the Agama lizard [Chang-Siu et al., 2011; Libby et al., 2016a; Zhao et al., 2015]. Some robots employ tails for aiding more dynamic tasks similar to those perfected by the cheetah, including accelerating and decelerating [Patel and Braae, 2014] or turning [Casarez et al., 2013; Kohut et al., 2013a; Patel and Braae, 2013]. Tails have also been used in other platforms as the primary mechanism for injecting energy into a gait rather than simply augmenting locomotion [Balasubramanian et al., 2008; Berenguer and Monasterio-Huelin, 2008; De and Koditschek, 2015].

Despite these achievements, few robots are equipped with tails outside of research focused on the dynamics of tailed locomotion. This is largely due to the added mass and system complexity associated with the tail. Each of the robots highlighted above employ their tails for inertial reorientation [Libby et al., 2016a] – leveraging the high inertia of the tail to allow the motor to do work to reorient the body before the tail exhausts its range of motion. This means that effective tails must have high inertia, which often does not comply with the tight payload budgets of mobile robots. In contrast, cheetah tails weigh roughly 2% of their body mass, with most of the mass in muscle at the base of the tail resulting in low inertia [Hudson, 2011; Patel et al., 2016].

One hypothesis for the utility of cheetah tails suggests they perform aerodynamic rather than inertial reorientation [Patel et al., 2016]. Most mammal tails are covered in fur, including those shown in Fig. 33, which increases their aerodynamic drag. When flicked or when moving through a steady airstream, the drag force resists the motion and allows for the application of wrenches to the body. Since this mechanism does not depend on mass, an aerodynamic drag tail can be extremely

Figure 33: Examples of biological and robotic systems with long aerodynamic tails. The cheetah, the giant Indian squirrel, and the jerboa have tails ranging from 75% to nearly 200% of their body length, motivating the aerodynamic tail presented here. Top left: Cheetah (*Acinonyx jubatus*). Bottom left: Greater Egyptian jerboa (*Jaculus orientalis*), photo credit: Elias Neideck/CC BY-SA 3.0. Middle: Giant Indian squirrel (*Ratufa indica*), photo credit: VinodBhattu/CC BY-SA 4.0. Right: The Ghost Robotics Minitaur [Kenneally et al., 2016], equipped with a 2x body length aerodynamic tail.

lightweight compared to an inertial tail. Aerodynamic drag tails have been shown to enable rapid turning on centimeter scale robots [Kohut et al., 2013b], although this turning was achieved by fixing the tail at an angle to act as a rudder, or dynamically actuated with all effects attributed to inertial reorientation rather than aerodynamic effects. To the knowledge of the authors, no other studies have investigated actuated aerodynamic effects or offered any comparative analysis for terrestrial tails.

This work investigates the extent to which aerodynamic drag affects the utility of tails in performing dynamic tasks. We present a model (Section 5.2) that captures the dynamics of aerodynamic tails. Then we use that model to show when aerodynamic drag is a significant contribution to control affordance compared to just inertial effects (Section 5.3). Leveraging this model, we constructed a tail (Section 5.4) to maximize aerodynamic drag while minimizing inertia, shown

in Fig. 33 along with several of the animals inspiring its design. Using this tail we demonstrate two biologically motivated behaviors on a robot, aerial self-righting and forward acceleration (Section 5.5) to show applications of this control affordance. Finally, we highlight the practical advantages aerodynamic drag tails exhibit over inertial tails (Section 5.6).

5.2 Aerodynamic Reorientation Model

Investigating the utility of aerodynamic drag forces in reorientation tasks requires a model to isolate and compare the aerodynamic and inertial effects. Such a model has been thoroughly researched for inertial reorientation [Libby et al., 2016a] but has not been developed for aerodynamic reorientation. The main result of this work is a model for aerodynamic reorientation and an accompanying metric for effectiveness, from which several insights into the dynamics and utility of aerodynamic drag tails are gleaned.

We model the tail as a non-porous rigid body moving at a relatively high velocity through a fluid, which applies a quadratic drag force [Landau and Lifschitz, 1959] according to the relationship

$$F_D = \frac{1}{2}\rho C_D A v^2 \tag{71}$$

where F_D is the aerodynamic drag force acting in the opposite direction of motion, ρ is the density of the fluid, C_D is the drag coefficient that captures the surface interaction between the fluid and the body, A is the surface area of the body, and v is the component of the velocity of the body orthogonal to the surface area.

The tail is pinned to a second rigid body (hereafter referred to as "the body") with no other external wrenches, such that the aerodynamic drag force applies a moment to the body. For simplicity we assume the tail velocity is purely determined by its angular velocity and that the aerodynamic drag force F_D is defined in the coordinate frame of the tail such that it is orthogonal to tail motion and therefore independent of the angular position of the tail. These modeling decisions serve to

isolate rotation and ignore any translational motion for the sake of clarity, although translational effects would benefit high speed systems as described in[Patel et al., 2016] and further discussed in Section 5.6. Integrating this moment along the length of the tail yields a net aerodynamic torque of

$$\begin{aligned} |\tau_D| &= \int_A F_D l \, dA \\ &= \int_{L_0}^{L} \frac{1}{2} \rho C_D w ((\dot{\theta}_b + \dot{\theta}_t) l)^2 l \, dl \\ &= \frac{1}{8} \rho C_D w (\dot{\theta}_b + \dot{\theta}_t)^2 (L^4 - L_0^4) \end{aligned} \tag{72}$$

where τ_D is the aerodynamic torque applied in the opposite direction of the tail rotation, w is the width of the tail (assumed constant over the length of the tail), L is the length of the tail, L_0 is the distance from the pin joint to the segment of the tail that generates drag, θ_b is the angle of the body with respect to the world frame, and θ_t is the angle of the tail with respect to the body. These parameters are shown in Fig. 34 for an example tail geometry, although this model can apply to other tail geometries so long as the corresponding drag coefficient is known.

This equation highlights the favorable scaling of aerodynamic drag tails, as the drag torque scales with L^4, or L^5 if tail width is assumed to scale with length. This favorable scaling is reflected in nature through animals like the cheetah, the giant Indian squirrel, or the jerboa, all of which have furry tails ranging from 75% to nearly twice body length, shown in Fig. 33 [Happold, 1967; Patel et al., 2016; Sushma and Singh, 2008]. Quartic length scaling underscores the importance of drag near the tail tip rather than the base, which is particularly notable in the jerboa's tuft of fur at the tip of its relatively thin tail. This equation also highlights favorable quadratic scaling with tail angular velocity, and linear scaling with width and drag coefficient. Mass and inertia are notably absent in this equation, suggesting that effective tails should be long yet lightweight and employed at high speeds. These factors are again consistent with many agile animals, particularly the cheetah [Patel and Braae, 2013].

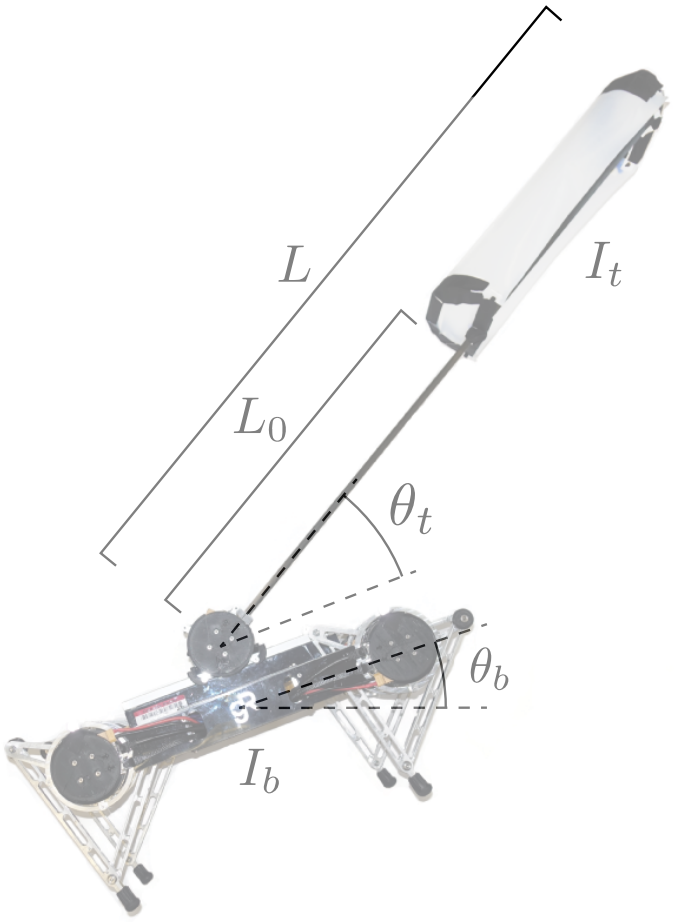


Figure 34: Schematic of tail system parameters. These parameters describe the tail geometry and model variables used to calculate effectiveness. Not labeled is tail width, w, which is into the page. The geometry of this particular tail is a half cylinder with diameter w, but the model in (72) is not restricted to any one tail shape.

5.3 Comparison of Inertial and Aerodynamic Effectiveness

Equation (72) highlights the important characteristics of aerodynamic drag tails, but does not immediately prescribe the magnitude of the control authority these tails provide. For purely inertial tails, this control authority has been quantified via tail effectiveness ξ, which is defined as the ratio of body rotation to tail rotation under the assumption of constant total angular momentum [Libby et al., 2016a]. For inertial tails, effectiveness is a straightforward function of the tail and body inertias. However, the nonlinearities of the aerodynamics in (72) necessitate numerical methods to quantify effectiveness for an aerodynamic drag tail. We investigate this by constructing the

equations of motion of the system in Fig. 34,

$$\begin{bmatrix} I_b + I_t & I_t \\ I_t & I_t \end{bmatrix} \begin{bmatrix} \ddot{\theta}_b \\ \ddot{\theta}_t \end{bmatrix} = \begin{bmatrix} \tau_D \\ G\tau + \tau_D \end{bmatrix} \tag{73}$$

$$\begin{bmatrix} \ddot{\theta}_b \\ \ddot{\theta}_t \end{bmatrix} = \begin{bmatrix} -\frac{G\tau}{I_b} \\ \frac{I_b+I_t}{I_b I_t} G\tau + \frac{1}{I_t}\tau_D \end{bmatrix} \tag{74}$$

where I_b and I_t are the body and tail inertias respectively and both defined with respect to the rotational joint, τ is the torque provided by the motor, and G is the gear ratio. We neglect the rotor reflected inertia since optimal gearing for the systems considered here result in reflected inertias much less than that of the tail. Numerically integrating these equations with initial conditions at rest yields the time evolution of the body and tail angles. Together, these quantities define the same tail effectiveness metric as in [Libby et al., 2016a] – the ratio of the body angle achieved to the tail angle swept – but here based on a particular behavior and control input.

We define a baseline task of aerial reorientation similar to that studied in [Libby et al., 2016a] which provides a particular measure of effectiveness. The task is to maximize body rotation in the time it would take for the system to fall one body length, motivated by recovering in mid-air from a fall off a ledge or a leap onto a surface. In addition to the task, aerodynamic tail effectiveness is a function of the gear ratio, actuator, and the tail geometry, as well as the body and tail inertias. To explore these relationships we select a body scale of a common quadrupedal robot, the Ghost Robotics Minitaur [Kenneally et al., 2016], and highlight 1x, 1.5x, and 2x body length tails. All tails are modeled as half cylinders with closed ends to maximize the drag coefficient [Hoerner, 1965], with width w equal to half of the robot body width. The parameters used for this simulation are shown in Table 4.

The equations of motion in (74) are integrated from rest with `ode45` in MATLAB R2018b to calculate the effectiveness metric. The tail is allowed to rotate freely for the specified duration or until $\theta_t = 180^\circ$ at which point the simulation is paused and a plastic impact is applied between the tail and the robot such that the tail is brought to rest and angular momentum is instantaneously

conserved. The system is then resumed until the end of the duration.

For each tail length, the optimal gear ratio is found by performing an integer line search, simulating the behavior with each gear ratio from one to 50 then choosing that which produced the highest resulting effectiveness. The torque τ is calculated at each instant with the following motor model:

$$i = \frac{V - k_t G \dot{\theta}_t}{R} \tag{75}$$

$$\tau = f(i) \tag{76}$$

$$f^{-1}(\tau) = 0.539\tau^3 + 8.93\tau \tag{77}$$

where i is the current through the motor armature, V is the battery voltage, k_t is the motor torque constant, R is the resistance of the motor, and $f(i)$ is a function that maps current to torque. The values of these parameters are shown in Table 4. The inverse of f shown in (77) is obtained by fitting a cubic polynomial to the empirical torque-current relationship given in [De and Koditschek, 2015] for the motor (T-motor U8 KV100). This nonlinearity is included to approximate the significant magnetic saturation these motors experience at high currents. This motor model reflects the chosen platform but also captures the general torque-speed relationship of a DC motor.

Specifying the body inertia, tail geometry, and actuation model along with the optimal gear ratio allows for exploration of the relative effect of tail inertia and aerodynamics on effectiveness. We normalize the tail inertia I_t by the total inertia $I_b + I_t$ and sweep across a range of inertias to find the resulting body angle displacements. For each inertia, the effectiveness is calculated both with and without aerodynamic drag to obtain the pure inertial effectiveness and the combined effectiveness, such that the difference between the two defines the aerodynamic contribution. Fig. 35 shows these results.

At low tail inertia, most of the effectiveness of the tail comes from aerodynamic drag. As the inertia increases, the the impact caused by the finite range of motion of the tail increases enough to decrease overall effectiveness – this is quite different from inertial tails that always benefit from

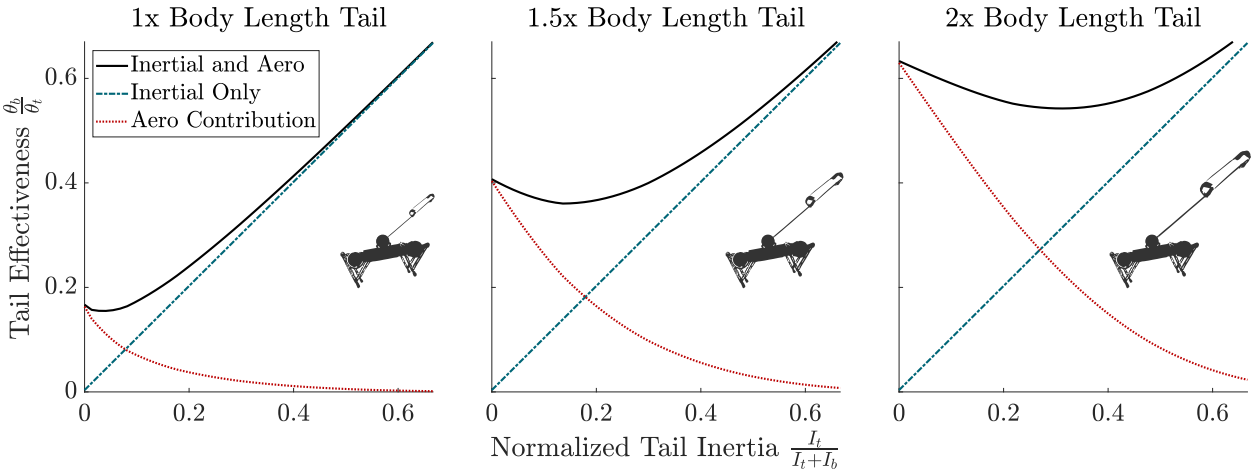


Figure 35: Aerodynamic effectiveness for various tail lengths and inertias. Aerodynamic drag yields highly effective tails for low inertia. When aerodynamics are not considered, effectiveness (dash-dotted line) is a function of only normalized tail inertia and independent of tail geometry. When enabled, effectiveness of the tail increases significantly (solid line). The aerodynamic component (dotted line) is equal to the total effectiveness minus the inertial effectiveness. Each plot indicates the tail length used to calculate effectiveness.

increasing inertia. As the inertia increases further, the tail acceleration and velocity decrease which allows for larger torques to be applied for a longer duration. This effect eventually outweighs the reduced aerodynamic component and the induced impact cost, causing the total effectiveness to increase again but only for significantly higher inertias. A purely inertial tail would require almost twice the inertia of the body itself to match the effectiveness of the longest massless aerodynamic drag tail.

The importance of tail length in aerodynamic effectiveness is highlighted in Fig. 35. The short tail maintains a notable improvement in effectiveness for relatively low inertias, but the long tail maintains a significant margin of improvement for a wide range of inertias, yielding a 50% or greater improvement in effectiveness up to a tail normalized inertia of 0.37, or roughly half the body inertia. The long tail is almost four times as effective as the short tail in the massless case. This underscores the importance of length over mass in aerodynamic tail design. Both lighter and longer tails allow for increased tail tip velocities and improved effectiveness. This is a favorable trend for both biology and robotics, as lighter appendages allow for more agile behaviors and increase the allowable payload of the system.

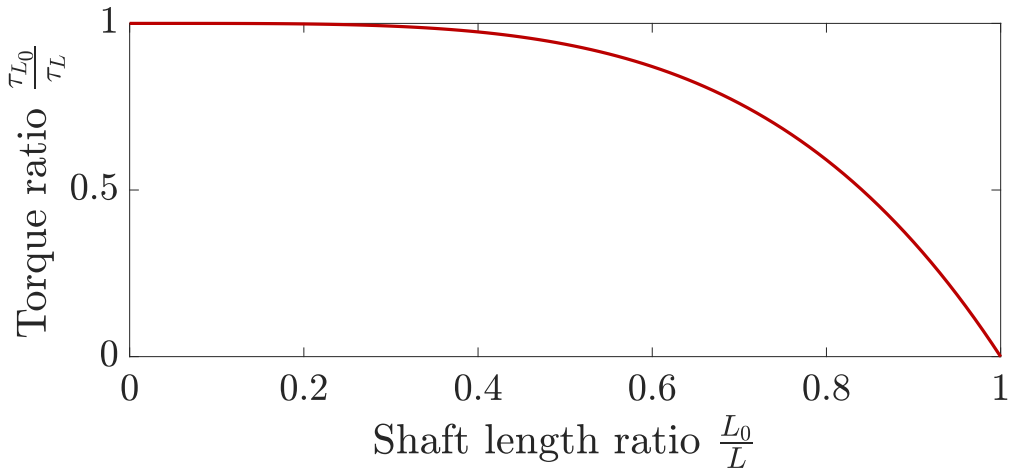


Figure 36: The effect of tail shaft length on the aerodynamic drag torque for a constant overall tail length L. The applied torque decreases with increasing tail shaft length L_0, but the quartic scaling results in a significant reduction only for values of L_0 close to L. The x-axis here shows the ratio of the tail shaft L_0 to the total length L, and is equivalent to the fraction of the tail that produces no aerodynamic drag. The y-axis describes the ratio of the torque τ_{L_0} produced by a tail of length L with shaft length L_0 to the torque τ_L produced by a tail of length L with $L_0 = 0$.

Other properties of the tail such as width w, tail shaft length L_0, and the tail geometry (i.e. the corresponding drag coefficient) all affect aerodynamic effectiveness in addition to tail length. Equation (72) also exposes the effects of these additional properties. Aerodynamic torque scales linearly with width and drag coefficient, so these quantities should be maximized subject to relevant design constraints. Tail shaft length is optimal at $L_0 = 0$ m, but due to its quartic scaling the aerodynamic torque is only significantly reduced for values of L_0 close to L, as shown in Fig. 36. Other design factors such as the weight of the tail material or the tail rigidity may encourage larger values of L_0, so a designer must strike a balance between these design choices and optimality.

5.4 Hardware Implementation

Verifying the model results above in hardware requires a highly effective tail to maximize drag. As previously discussed, aerodynamic effectiveness is dependent on tail geometry and in particular the drag coefficient, C_d. The drag coefficient is a dimensionless measure of resistance of a body to transverse fluid flow. In biological tails, fur increases this resistance substantially without adding significant mass to the tail [Patel et al., 2016]. Engineered tails have more flexibility in their design,

and can employ other geometries and materials to maximize drag and minimize mass.

The tail constructed for the aerial self-righting and forward acceleration tasks presented here features a 1 m long (2x body length), 1 cm diameter carbon fiber shaft (2159T85 McMaster-Carr) fitted with a 18 cm wide UHMW polyethylene film (85655K13 McMaster-Carr) half cylinder scoop at the end. These materials were selected for their high strength-to-weight ratios. The ends of the scoop were sealed to prevent air from escaping radially. The whole tail weighs 110 g with an inertia of 0.058 kg·m^2, which when normalized for the tested hardware platform yields a normalized tail inertia of 0.204. The tail is rigidly mounted to the output of a planetary geartrain driven by a U8 motor, which in turn is fixed to the robot chassis. The optimality of the gear ratio was determined by the process outlined in Section 5.3. These and other hardware parameters are listed in Table 4, and the tail can be seen in Figs. 33 and 34.

It should be reiterated that these values for L, L_0, and w are not optimal for aerodynamic effectiveness, as optimality would be achieved with $L = \infty$, $L_0 = 0$, and $w = \infty$. These parameters are bounded by other design constraints which a roboticist may select for a particular application. In this case we prioritize a reasonable motion envelope and tail rigidity, and therefore restrict the tail length as inspired by the animals in Fig. 33, and the tail width and shaft length to maintain rigidity in the UHMW film.

To calculate the drag coefficient of this tail, a smaller tail with the same shape was fixed to the motor used in the above experiments, and spun freely with different voltages to produce steady state angular velocities. The resulting aerodynamic torque was then calculated by measuring the current and voltage supplied to the motor and equating the electrical power input to the sum of the resistive losses in the motor and the mechanical power output to yield the aerodynamic torque. This resulted in a torque-angular velocity curve which was fit to a quadratic as shown in Fig. 37, with $R^2 = 0.977$ indicating good quadratic fit. Matching this curve to that in (72) and accounting for the tail dimensions yields a drag coefficient of 2.0, slightly less than the theoretical value of 2.3 [Hoerner, 1965].

Table 4: Minitaur, Tail, and Actuator Parameters.

Parameter	Symbol	Value	Units
Body mass	–	7.3	kg
Body length	–	0.50	m
Body inertia	I_b	0.23	kg·m^2
Tail framing mass	–	0.706	kg
Tail mass	–	0.110	kg
Tail length	L	1.0	m
Tail shaft length	L_0	0.60	m
Tail width	w	0.18	m
Tail inertia	I_t	0.058	kg·m^2
Tail normalized inertia	$\frac{I_t}{I_t+I_b}$	0.204	-
Tail gear ratio	G	4:1	-
Tail drag coefficient	C_D	2.0	-
Motor torque constant	k_t	0.0954	$\frac{\text{N·m}}{\text{A}}$
Motor winding resistance	R	0.186	Ω
Motor voltage	V	16	V

5.5 Experimental Results

To verify these model results and highlight practical applications for aerodynamic reorientation, we demonstrate two tail-assisted tasks: aerial self-righting and forward acceleration. The aerial self-righting task corresponds to the model analysis and shows the utility of expanded control authority, whereas the forward acceleration task tests the ability of the tail to apply this control authority to a dynamic task that requires both linear and rotational motion.

5.5.1 Aerial Self-Righting

To verify these model results and highlight practical applications for aerodynamic reorientation, we first demonstrate the use of an aerodynamic drag tail in an aerial self-righting task. This task corresponds directly to the model analysis and shows both the utility and the magnitude of expanded control authority such a tail offers.

The aerial self-righting task was executed by dropping the system from a height and providing a step input to the actuator of a 2x body length tail so that the system rotates 90 degrees and lands on its feet. This task was previously tested for robots with inertial tails in [Libby et al., 2012,

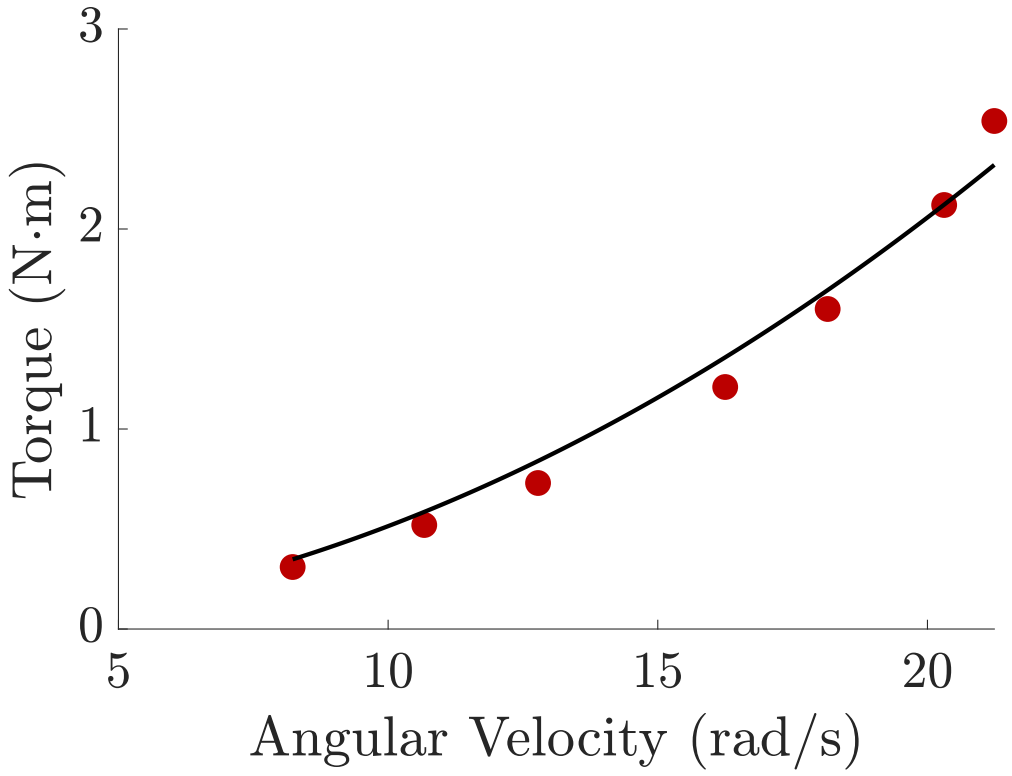


Figure 37: Experimental validation of drag coefficient and quadratic torque to velocity relationship. Aerodynamic torque produced by a half cylinder aerodynamic drag tail was measured while rotating the tail at various angular velocities. The red dots represent experimental data, and the black line is a quadratic fit given by $\tau_D = 0.00514\,\dot{\theta}_t^2$, with an $R^2 = 0.977$ indicating good fit.

2016a], which were able to successfully land on their feet but required a tail with more inertia than the body to do so. This experiment directly tests the ability of a system to aggressively reorient, as failure to reorient quickly can result in severe damage to the system if it cannot land on its feet.

To perform this experiment, a Ghost Robotics Minitaur was equipped with the previously described aerodynamic drag tail, oriented vertically and dropped from a height of one body length (0.5 m). The drop height was defined as the vertical displacement of the center of mass from the beginning of the drop to the final resting position. The robot was held aloft by a quick release clip activated by pulling a pin that holds the clip in place. This pin was pulled at the same time that the robot was commanded to begin the reorientation behavior. High speed camera footage confirmed that the robot consistently began the reorientation a few milliseconds after the clip was released. Body pitch data was recorded with an Optitrack motion capture system. The resulting

pitch trajectory is shown in Fig. 38, and a time sequence of the behavior is shown in Fig. 39. The simulation data in Fig. 38 was synchronized with the experimental data at the last instant before the motion capture data showed a body pitch displacement. This instant occurs a few milliseconds after the step input is provided at $t = 0$ due to backlash in the geartrain and tail deflection. This experiment consisted of four trials to reduce any noise in the data, although Fig. 38 shows that the standard deviation between these trials was very small ($\sigma = 1.5^\circ$ on average over the course of the fall).

The robot tracks the model-predicted trajectory well, rotating 90 degrees in 302 ms on average (standard deviation of 4 ms). The slight delay in body pitch tracking is likely due to the deflection in the carbon fiber tail shaft, which slows the acceleration of the tail. This rapid reorientation allows the robot to land safely and absorb the impact with its legs rather than its chassis. Without the tail, the robot simply falls straight to the ground, which for falls exceeding a body length can easily damage the robot. Notably, this resulting effectiveness matches that of the inertial tail tested in [Libby et al., 2016a], but with a normalized tail inertia of 0.204 rather than 0.558, a reduction of 63%. A tail of the same inertia as tested but with no drag was simulated, with the resulting trajectory in Fig. 38 indicating only 37 degrees of rotation. Note that in the purely inertial case this effectiveness can be calculated directly from the normalized inertia using the relationship derived in [Libby et al., 2016a] without integration.

5.5.2 Forward Acceleration

Lightweight aerodynamic tails are effective at reorientation tasks, but successful biological or robotic locomotion involves a broad range of agile behaviors, such as running, jumping, or turning. Often these behaviors require precise application of large forces into the ground in order to accelerate the body in a desired direction. Equipping a robot with a tail adds mass to the system, which reduces the acceleration a given force produces. However, aerodynamic drag tails provide additional control authority which could be employed to increase the agility of the system.

We leverage trajectory optimization to investigate the effect of aerodynamic drag tails on the

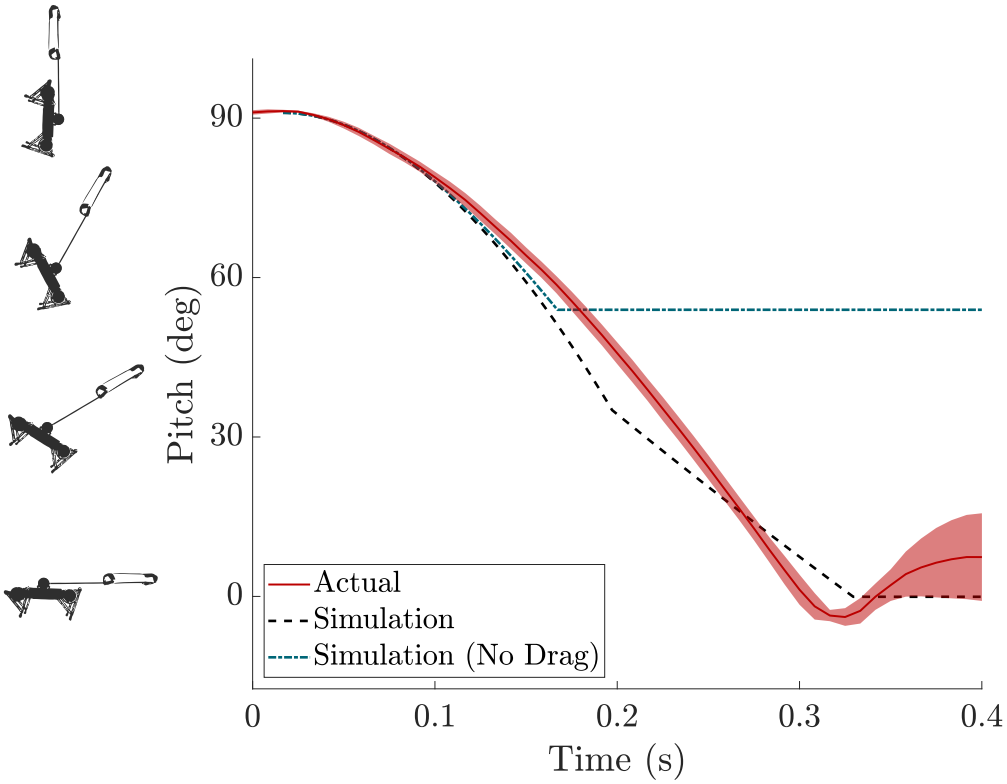


Figure 38: Aerial self-righting pitch trajectory. Motion capture orientation data during the aerial-self righting task (solid line) show the robot tracking the model predicted trajectory (dashed line) until ground impact, with a slight delay due to unmodeled tail shaft deflection. A simulation of the tail with no aerodynamic drag is included for comparison (dash-dotted line). The discontinuity in the velocity of each tail simulation is caused by the modeled plastic impact between the tail and the robot body, which is significantly more problematic without aerodynamic effects. This discontinuity is not observed in the experimental data as the shaft deflection delayed this impact until just before touchdown and reduced its plasticity. The shaded region represents 1σ variance (N=4).

forward acceleration of a legged robot. This method represents the motion of the robot with a parameterized trajectory, defines the desired task through constraint functions evaluated on that trajectory, and determines optimality with respect to a cost function. This is a well-studied method in optimal control so we refer to prior literature for details [Nguyen et al., 2019; Schultz and Mombaur, 2009; Witkin and Kass, 1988].

The task encoded here is very similar to the quadruped leaping behavior described in [Nguyen et al., 2019], and shares the same constraints on dynamic feasibility, contact schedule, joint and torque limits, friction cones, and initial resting configuration. Unlike [Nguyen et al., 2019], we op-

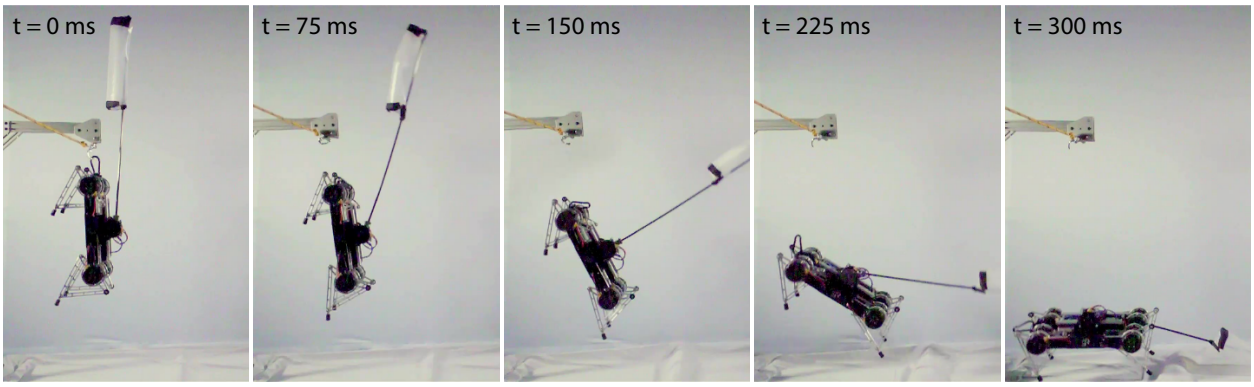


Figure 39: Aerial self-righting time sequence. The robot rotates from vertical to horizontal in one body length of travel. The first frame shows the moment the robot was released, and the last frame shows the feet of the robot just above the ground. The second frame shows the deflection visible in the tail shaft – despite the deviations from the model induced by this deflection, the robot is still able to rotate a full 90 degrees in one body length of free fall.

timize for the forward velocity of the system at liftoff divided by the behavior duration to maximize acceleration. We do not constrain the duration of each contact phase as in [Nguyen et al., 2019], although we do apply a lower bound of 1.3 m/s – the average running speed of the robot – to the final forward velocity to replicate a stand-to-run gait transition. Dynamic and kinematic constraint bounds are derived from the Minitaur robot parameters in Table 4, the motor model in (75)–(77), and [Kenneally et al., 2016]. Separate trajectories are optimized both with and without a tail, which starts at rest. The trajectories are transcribed into a direct collocation hybrid trajectory optimization framework in FROST [Hereid and Ames, 2017a] and solved with IPOPT [Biegler and Zavala, 2009]. Each optimization is seeded with an initial trajectory of the average of the upper and lower bounds on each variable.

The outputs of this process are time-parameterized trajectories of the robot state, joint trajectories and torques (for both the leg and tail motors), and contact forces. These trajectories were tracked on the hardware by replaying the open loop joint torques and applying joint level PD feedback to track the trajectory of each individual joint. Each trajectory was executed six times, and the resulting velocity of each trajectory was measured by recording position with an OptiTrack motion capture system and differentiating the signal with respect to time. The final velocity was defined as the forward velocity of the system after the feet left the ground, and the duration of the behavior

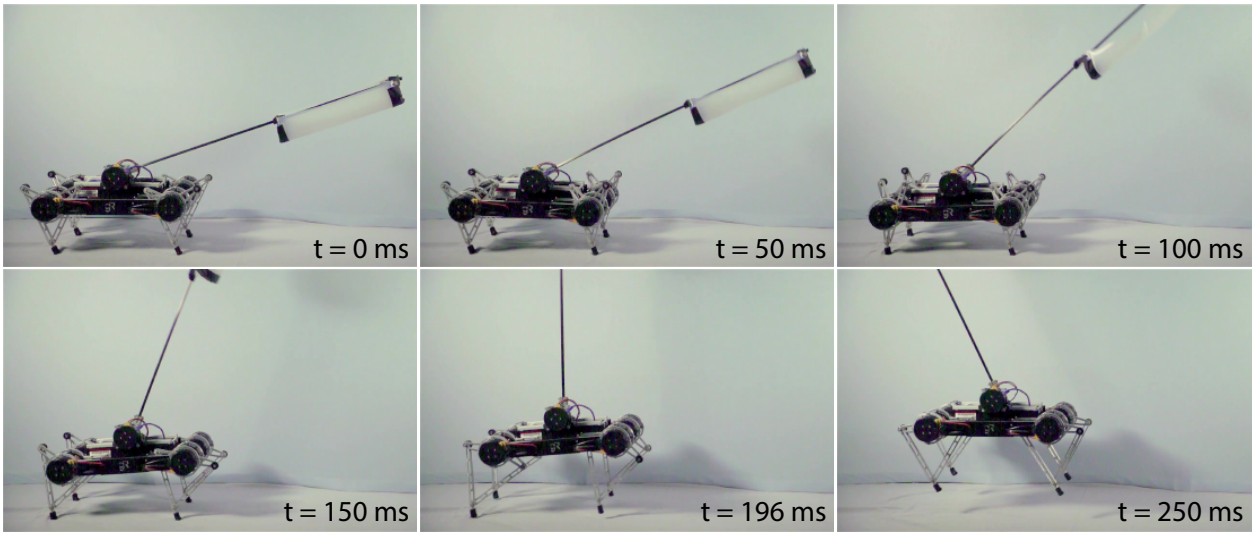


Figure 40: Forward acceleration behavior time sequence. The time sequence of the acceleration behavior shows the tail swinging backwards, shifting the robot forwards into a position to extend its legs. The behavior ends as the robot's feet lift of the ground at 196 ms. The final frame shows the forward motion after liftoff, although this is not considered in calculating average acceleration.

was given by the trajectory. Prior to each round of testing, the battery was charged to full voltage and the motors were allowed to cool to ensure external conditions remained consistent. The time sequence of the behavior is shown in Fig. 40, and the resulting velocity trajectories and average accelerations are shown in Fig. 41.

With the long aerodynamic tail, the robot accelerates at 6.3 m/s^2, 12% faster than the 5.6 m/s^2 without a tail despite a 10% increase in robot mass. The robot accelerates 18% faster than the 5.3 m/s^2 if the tail were installed but inactive. This increase is achieved by swinging the tail backwards as the legs prepare for and then execute the forward leap, aided in part by a small amount of forward impulse exerted on the tail (5% of the total change in momentum of the system). Swinging the tail backwards provides forward thrust on the robot, which allows the legs to extend earlier than without the tail, reducing the time required to reach top speed by 11%.

5.6 Discussion

The results presented above show that the utility of aerodynamic tails matches and in some cases exceeds that of inertial tails, and in doing so can provide a meaningful contribution to overall sys-

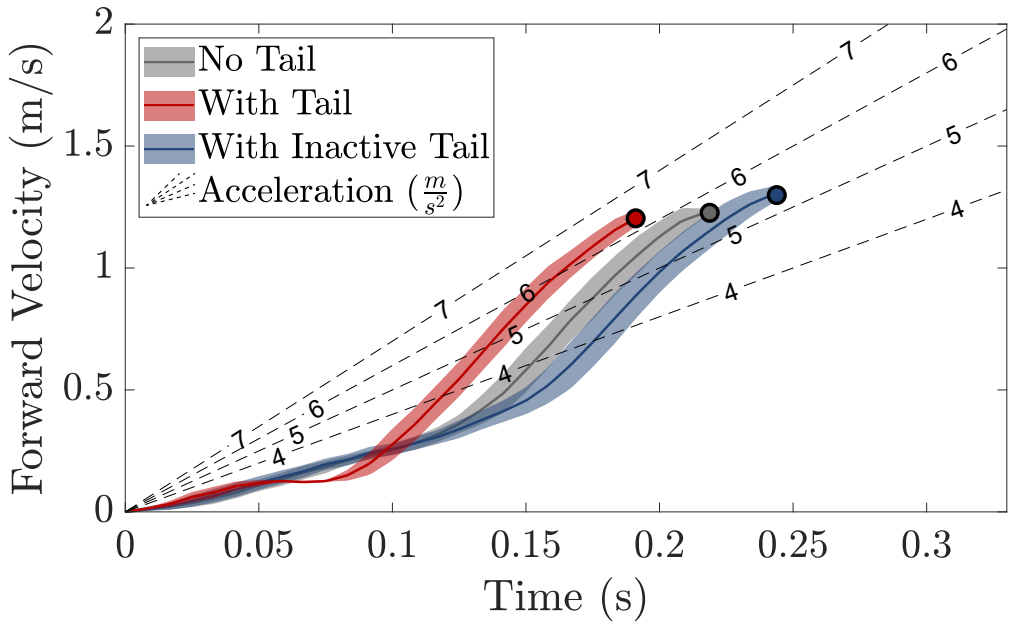


Figure 41: Forward acceleration behavior trajectories. Experimentally measured robot velocities over time show the robot reaching the same velocity faster with the tail. The shaded region represents 1σ variance (N=6). The dashed lines show the average acceleration to achieve a final velocity over a given duration.

tem agility. Aerodynamic tails also overcome several shortcomings that often plague inertial tails. Based on the quantitative results shown above and from our experience working with aerodynamic tails, we highlight a few of these key properties.

Mass and Inertia

The most evident advantage of aerodynamic tails is their low mass. Since producing aerodynamic drag is independent of mass (unlike inertial reorientation), aerodynamic tails can be extremely lightweight. This increases the available payload of a system, and enables leg forces to produce higher accelerations. Lower tail inertia also reduces the impulse required to arrest tail motion. Inertial tails are capable of rejecting disturbances rapidly by transmitting energy into the tail, but shortly thereafter this energy must be removed, either by applying a counter-torque to slow the tail or from an impact between the tail and the body. This presents a challenging planning problem to precisely regulate the flow of energy from the tail to the body over time. Aerodynamic tails mitigate this issue by providing large amounts of control authority for significantly less tail momentum.

Lower mass also reduces the gravitational moment on the tail, requiring less torque to hold the tail in a static position when active control is not required.

External Work

Aerodynamic tails also have a distinct advantage over inertial tails in that they can do work on the environment. This ability allows them to apply continuous torque even at zero acceleration, unlike inertial tails which produce no torque at zero acceleration. This ability also enables non-zero net impulses. Since inertial tails transfer momentum internally between the tail and the body, once the tail sweeps its full range of motion both tail and body return to their initial angular velocity. Lightweight aerodynamic tails can come to rest given a small internal impulse, but due to the net impulse from the environment the body can continue to rotate. This is apparent in the simulation data in Fig. 38. When the purely inertial tail collides with the body all rotation ceases, whereas the impact between the body and the aerodynamic tail (which occurs roughly 180 ms into the simulated self-righting behavior) is much smaller in magnitude, allowing further body rotation.

Effectiveness at High Speeds

Aerodynamic tails also benefit from favorable scaling at high speeds. Since both the aerodynamic force and generated moment scale with velocity squared, any tail motions executed while the robot is moving rapidly are amplified. The model derived in (72) can be adjusted to account for these translational effects, producing

$$\begin{aligned} |\tau_D| &= \int_A F_D l \, dA \\ &= \int_{L_0}^{L} \frac{1}{2}\rho C_D w((\dot{\theta}_b + \dot{\theta}_t)l + \dots \\ &\qquad \cdots - v_x \sin(\theta_b + \theta_t) + v_y \cos(\theta_b + \theta_t))^2 l \, dl, \end{aligned} \tag{78}$$

where v_x and v_y are the components of the body velocity expressed in the same spatial coordinate frame that defines body rotation. We omit the closed-form solution of (78) due to its length, but

the integrand still highlights how the applied torque scales quadratically with the linear velocity of the body. This phenomenon has been explored for cheetahs in [Patel et al., 2016], which showed a 28% improvement to angular impulse while turning at 30 m/s compared to a static airstream. This is useful for legged systems as the high speeds that enable this expanded control authority often require larger magnitudes of actuation for stability.

Scaling with Body Length

The scaling analysis offered here has largely been focused on the scaling of the tail length for a fixed body length to highlight the importance of a long and lightweight tail. Isometrically scaling the tail with the body length would also change the tail width, resulting in an aerodynamic torque that scales with L^5. Interestingly, this matches the scaling of inertia with body length, as inertia scales with mL^2, where m is the mass of the system and scales with L^3. This differs from the conclusion in [Kohut et al., 2013b] that aerodynamic drag scales with L^3 – here we assume that the tail velocity is determined by angular rotation rather than simply forward velocity. This suggests that aerodynamic drag tails can be effective at any scale, although the additional translational effects may be primarily useful at smaller scales, provided that the Reynolds number remains high enough to induce turbulent flow.

Limitations

The favorable scaling of aerodynamic drag tails does not come without limitation. Often robots that operate in close quarters have tight restrictions on their workspace, and for such systems a body length or longer tail may be infeasible. Aerodynamic tails also benefit from length and low mass, which induces a design tradeoff between effectiveness and durability – a highly effective tail may not be able to withstand large interaction forces with the environment. These remain open design challenges, although many biological systems employ lightweight, flexible structures to reduce impacts, modify the tail workspace, and increase durability. It should also be emphasized that studying the nonlinear nature of aerodynamic drag requires a prespecified task, actuation model,

geometry, and scale, all of which affect the utility of the tail. Despite an effort to isolate each of these components and offer insight into each in turn, the synthesis of these components remains a behavior-specific challenge.

5.7 Conclusions

The favorable scaling of aerodynamic tails make them lightweight yet effective tools to increase the control affordance of a system. This paper presents a model and a corresponding metric to analyze these aerodynamic effects, shows that the magnitude of control affordance can be substantial for long and lightweight tails, and demonstrates this utility in two dynamic behaviors on relevant hardware. A roboticist seeking to overcome underactuation of a system without adding large amounts of mass can thus employ an aerodynamic drag tail by maximizing the components of (72) subject to applicable design constraints. The ability of the aerodynamic drag tail to exert forces directly on the environment coupled with its low momentum compared to the body also makes planning stable motions much simpler. Together, these factors improve the practical implementation of the tail, enabling its application to a wide variety of systems where control authority is critical.

6 Conclusion

Attaining autonomous and agile legged locomotion in unstructured terrain remains a difficult task due to the complicated underlying dynamics and kinematics. A robot tasked with point-to-point navigation must consider which route to take to get to the goal, where to step in order to move as desired, and apply the right forces to the ground to execute that motion, all while rejecting disturbances and handling inaccurate and delayed information. Planning the entire motion all at once with full-order expressions for the kinematic and dynamic constraints is computationally infeasible and prone to yielding highly myopic and brittle behavior. Likewise, locomoting blindly and relying on reactive behaviors or simple models quickly fails when confronted with infeasible regions of the state space.

This work has presented novel methods to handle these challenges by adapting the complexity of the solution to the task. The hierarchical planning and control framework presented in Section 3 introduces a breakdown of the motion planning problem with an explicit focus on agility and autonomy through its support of high kinetic energy motion, terrain awareness, and real-time operation. The high-level motion planner presented in this work provides the system with real-time routes through the terrain that account for the capabilities of the platform and guide the system towards the goal, and several demonstrations of this system showed both its efficacy in planning long horizon behaviors and executing agile motions. The model predictive control formulation presented in Section 4 provided a method for establishing communication between layers of such a hierarchy by adapting the model used for predictive control to account for constraints in the higher-dimensional system. This method was shown to be provably stable provided conditions regarding a template and anchor relationship were met, and its performance improvement over fixed complexity formulations was demonstrated over candidate environments for a quadrupedal system. This work also presents a novel aerodynamic tail design that can be employed to expand control authority and reject disturbances when leg planning and control are insufficient. The presented design greatly decreases the mass and improves the practicality of such tails, rendering them more favorable for deployment.

These methods directly improve the utility of legged robots in industrial settings. By improving the speed and range of terrain over which the legged robot can navigate, these platforms can carry out applications more quickly and reliably. These qualities will enable their use in a wider range of applications, particularly those with unstructured terrain where other mobile robots struggle. One particularly promising domain for legged robotics is environmental monitoring. The author has directly contributed to work on robotic soil sampling for contaminant mapping and remediation, which often must be conducted in environments where large obstacles and uneven terrain block the passage of wheeled or tracked platforms. Despite the potential of legged platforms to succeed in these environments, a tracked platform was selected due to concerns over the reliability and autonomy of legged platforms in traversing such terrain. Successful deployment of the methods presented here could permit the usage of legged platforms and provide access to a much larger set of data. This trend could apply similarly to other applications such as delivery, inspection, public health, and space exploration.

It should be reiterated that many of the strategies employed to achieve real-time algorithm performance rely on heuristics and model reductions specific to legged locomotion. It is an intentional decision to restrict the domain of interest to this specific class of systems rather than generalizing to other forms of multi-contact dynamical systems such as those commonly found in manipulation. This framework could theoretically be generalized to apply to such systems, but that generalization remains outside the scope of this work as many of the heuristics introduced here would no longer apply. Similarly, aerodynamic tail designs are investigated specifically with relation to reorientation tasks for locomotion. Although they could be deployed in other systems where stabilizing torques are desirable, we leave their application to such domains for future work.

Although the methods presented here show technological advances in planning and control, there is still much progress to be made in fully realizing the potential of legged robots. Environments with extremely tight tolerances on acceptable error or allowable configurations will likely not support the model reductions or approximations presented here and will likely require more rigorous notions of feasibility, or employ learning-based approaches which are able to handle con-

straints in a robust way without learning overly conservative behaviors. Such improvements would increase the range with which navigation could be planned and the quality of the resulting motion. In a similar vein, advancements that relax the physical constraints on possible robot motions – such as expanding the friction cone through intelligent foot design or increasing actuation limits either through better actuators or transmission mechanisms – would aid any motion generation. Finally, this work has focused on improving performance in point-to-point navigation tasks, but those waypoints still must be specified by a user and over a terrain with which it understands how to interact. To truly unlock legged robot locomotion as a force for automation applications, robots much be able to adapt to entirely new scenarios while also making high-level decisions about how to best execute the task at hand.

References

Agility Robotics. Robots — Agility Robotics, 2020. URL `https://www.agilityrobotics.com/robots{#}digit`.

R. M. Alexander. The gaits of bipedal and quadrupedal animals. *The International Journal of Robotics Research*, 3(2):49–59, 1984.

R. M. Alexander. Principles of animal locomotion. In *Principles of Animal Locomotion*. Princeton University Press, 2013.

R. M. N. Alexander. The mechanics and energetics of hopping by kangaroos (Macropodidae). *J. Zool.*, 177:265–303, 1975.

F. Allgöwer and A. Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

C. J. Arellano and R. Kram. The effects of step width and arm swing on energetic cost and lateral balance during running. *J. Biomech.*, 44(7):1291–1295, 2011. ISSN 00219290. doi: 10.1016/j.jbiomech.2011.01.002.

C. G. Atkeson, P. W. Benzun, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padir, F. Polido, G. G. Tighe, and X. Xinjilefu. What happened at the DARPA robotics challenge finals. *Springer Tracts Adv. Robot.*, 121:667–684, 2018. ISSN 1610742X. doi: 10.1007/978-3-319-74666-1_17.

H. Audren, J. Vaillant, A. Kheddar, A. Escande, K. Kaneko, and E. Yoshida. Model preview control in multi-contact motion-application to a humanoid robot. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 4030–4035, 2014.

R. Balasubramanian, A. A. Rizzi, and M. T. Mason. Legless locomotion: A novel locomotion technique for legged robots. *Int. J. Rob. Res.*, 27(5):575–594, 2008. ISSN 02783649. doi: 10.1177/0278364908091023.

S. Bartoszyk, P. Kasprzak, and D. Belter. Terrain-aware motion planning for a walking robot. In *IEEE Intl. Workshop on Robot Motion and Control*, pages 29–34, 2017.

I. Batkovic, M. Ali, P. Falcone, and M. Zanon. Model predictive control with infeasible reference trajectories. *arXiv preprint arXiv:2109.04846*, 2021.

S. Bazeille, V. Barasuol, M. Focchi, I. Havoutis, M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini. Quadruped robot trotting over irregular terrain assisted by stereo-vision. *Intelligent Service Robotics*, 7(2):67–77, 2014.

J. M. Beer, A. D. Fisk, and W. A. Rogers. Toward a framework for levels of robot autonomy in human-robot interaction. *Journal of human-robot interaction*, 3(2):74–99, 2014.

C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter. Advances in real-world applications for legged robots. *Journal of Field Robotics*, 35(8):1311–1326, 2018.

A. Bemporad and M. Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 1999.

F. J. Berenguer and F. M. Monasterio-Huelin. Zappa, a quassi-passive biped walking robot with a tail: Modeling, behavior, and kinematic estimation using accelerometers. *IEEE Trans. Ind. Electron.*, 55(9):3281–3289, 2008. ISSN 02780046. doi: 10.1109/TIE.2008.927982.

P. A. Bhounsule, J. Cortell, and A. Ruina. Design and control of ranger: an energy-efficient, dynamic walking robot. In *Adaptive Mobile Robotics*, pages 441–448. World Scientific.

L. T. Biegler and V. M. Zavala. Large-scale nonlinear programming using IPOPT: An integrating

framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33 (3):575–582, 2009.

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

G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim. MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 2245–2252, 2018.

R. Blickhan. The spring-mass model for running and hopping. *Journal of biomechanics*, 22(11-12):1217–1227, 1989.

M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart. State estimation for legged robots-consistent fusion of leg kinematics and IMU. In *Robotics: Science and Systems*, volume 17, pages 17–24. MIT Press, 2013.

Boston Dynamics. Spot® — Boston Dynamics, 2020. URL `https://www.bostondynamics.com/spot`.

M. Brandao, M. Fallon, and I. Havoutis. Multi-controller multi-objective locomotion planning for legged robots. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4714–4721. IEEE, 2019.

D. Brandt. Comparison of A* and RRT-Connect motion planning techniques for self-reconfiguration planning. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 892–897, 2006.

R. Briggs, J. Lee, M. Haberland, and S. Kim. Tails in biomimetic design: Analysis, simulation, and experiment. *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 1473–1480, 2012. ISSN 21530858. doi: 10.1109/IROS.2012.6386240.

M. Brunner, B. Brüggemann, and D. Schulz. Hierarchical rough terrain motion planning using an optimal sampling-based method. In *IEEE Intl. Conference on Robotics and Automation*, pages 5539–5544, 2013.

K. Byl, A. Shkolnik, S. Prentice, N. Roy, and R. Tedrake. Reliable dynamic motions for a stiff quadruped. In *Experimental robotics*, pages 319–328. Springer, 2009.

S. Caron, Q.-C. Pham, and Y. Nakamura. ZMP support areas for multicontact mobility under frictional constraints. *IEEE Transactions on Robotics*, 33(1):67–80, 2016.

J. Carpentier and N. Mansard. Multicontact locomotion of legged robots. *IEEE Transactions on Robotics*, 34(6):1441–1460, 2018.

J. Carpentier, R. Budhiraja, and N. Mansard. Learning feasibility constraints for multi-contact locomotion of legged robots. In *Robotics: Science and Systems*, page 9p, 2017.

C. Casarez, I. Penskiy, and S. Bergbreiter. Using an inertial tail for rapid turns on a miniature legged robot. *IEEE Int. Conf. Robot. Autom.*, pages 5469–5474, 2013. ISSN 10504729. doi: 10.1109/ICRA.2013.6631361.

E. Chang-Siu, T. Libby, M. Tomizuka, and R. J. Full. A lizard-inspired active tail enables rapid maneuvers and dynamic stabilization in a terrestrial robot. *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 1887–1894, 2011. doi: 10.1109/IROS.2011.6048342.

M. Chignoli and S. Kim. Online trajectory optimization for dynamic aerial motions of a quadruped robot. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7693–7699. IEEE, 2021.

M. Chignoli and P. M. Wensing. Variational-based optimal control of underactuated balancing for dynamic quadrupeds. *IEEE Access*, 8:49785–49797, 2020.

S. L. Cleac'h, T. Howell, M. Schwager, and Z. Manchester. Fast contact-implicit model-predictive control. *arXiv preprint arXiv:2107.05616*, 2021.

D. Coleman, I. Sucan, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software: a MoveIt! case study. *arXiv preprint arXiv:1404.3785*, 2014.

S. H. Collins, M. Wisse, and A. Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001.

H. Dai, A. Valenzuela, and R. Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *IEEE Intl. Conference on Humanoid Robots*, pages 295–302, 2014.

A. De and D. E. Koditschek. The Penn Jerboa: A platform for exploring parallel composition of templates. *arXiv Prepr. arXiv1502.05347*, 2015.

R. Deits and R. Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *IEEE Intl. Conference on Humanoid Robots*, pages 279–286, 2014.

J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 1–9, 2018.

J. DiCarlo, B. Katz, P. Wensing, J. Ahn, and S. Kim. GitHub - mit-biomimetics/Cheetah-Software — github.com. `https://github.com/mit-biomimetics/Cheetah-Software`, 2019. [Accessed 11-Apr-2022].

Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park. Representation-free model predictive control for dynamic motions in quadrupeds. *IEEE Transactions on Robotics*, 37(4):1154–1171, 2021.

J. M. Donelan, S. M. O 'connor, T. J. Dawson, and R. Kram. The kangaroo's tail propels and powers pentapedal locomotion. *Biol. Lett.*, 10(figure 1):1–4, 2014. doi: 10.1098/rsbl.2014.0381.

A. Dornbush, K. Vijayakumar, S. Bardapurkar, F. Islam, M. Ito, and M. Likhachev. A single-planner approach to multi-modal humanoid mobility. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4334–4341. IEEE, 2018.

J. M. Duperret, G. D. Kenneally, J. Pusey, and D. E. Koditschek. Towards a comparative measure of legged agility. In *Experimental Robotics*, pages 3–16. Springer, 2016.

P. Eckert and A. J. Ijspeert. Benchmarking agility for multilegged terrestrial robots. *IEEE Transactions on Robotics*, 35(2):529–535, 2019.

D. Faconti. PlotJuggler. `https://www.plotjuggler.io/`. [Accessed 23-Apr-2022].

P. Fankhauser and M. Hutter. A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In A. Koubaa, editor, *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, chapter 5. Springer, 2016. ISBN 978-3-319-26052-5. doi: 10.1007/978-3-319-26054-9{_}5.

P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *IEEE Intl. Conference on Robotics and Automation*, pages 1–8, 2018.

P. Fankhauser, S. Bachmann, D. Bellicoso, T. Bi, R. Diethelm, and C. Gehring. GitHub - leggedrobotics/free_gait: An Architecture for the Versatile Control of Legged Robots — github.com. `https://github.com/leggedrobotics/free_gait`, 2019. [Accessed 12-Apr-2022].

F. Farshidian, R. Grandia, and M. Spieler. GitHub - leggedrobotics/ocs2: Optimal Control for Switched Systems — github.com. `https://github.com/leggedrobotics/ocs2`, 2022. [Accessed 11-Apr-2022].

D. Ferguson and A. Stentz. Anytime RRTs. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 5369–5375, 2006.

P. Fernbach, S. Tonneau, A. Del Prete, and M. Taïx. A kinodynamic steering-method for legged multi-contact locomotion. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 3701–3707, 2017.

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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 387–394. IEEE, 2018.

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

M. Geisert, T. Yates, A. Orgen, P. Fernbach, and I. Havoutis. Contact planning for the ANYmal quadruped robot using an acyclic reachability-based planner. In *Annual Conference Towards Autonomous Robotic Systems*, pages 275–287. Springer, 2019.

Ghost Robotics. Legged UGVs — Ghost Robotics, 2020. URL `https://www.ghostrobotics.io/robots`.

S. Gilroy, D. Lau, L. Yang, E. Izaguirre, K. Biermayer, A. Xiao, M. Sun, A. Agrawal, J. Zeng, Z. Li, et al. Autonomous navigation for quadrupedal robots with optimized jumping through constrained obstacles. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 2132–2139. IEEE, 2021.

K. Gochev, B. Cohen, J. Butzke, A. Safonova, and M. Likhachev. Path planning with adaptive dimensionality. In *Fourth annual symposium on combinatorial search*, 2011.

K. Gochev, A. Safonova, and M. Likhachev. Planning with adaptive dimensionality for mobile manipulation. In *2012 IEEE International Conference on Robotics and Automation*, pages 2944–2951. IEEE, 2012.

D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing. Robotic vertical jumping agility via series-elastic power modulation. *Science Robotics*, 1(1), 2016.

D. W. Haldane, J. K. Yim, and R. S. Fearing. Repetitive extreme-acceleration (14-g) spatial jumping with Salto-1P. *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 3345–3351, 2017. ISSN 21530866. doi: 10.1109/IROS.2017.8206172.

D. Happold. Biology of the jerboa, Jaculus jaculus butleri (Rodentia, Dipodidae), in the Sudan. *Journal of Zoology*, 151(2):257–275, 1967.

K. Hauser and V. Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *IEEE Intl. Conference on Robotics and Automation*, pages 2493–2498, 2010.

K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox. Motion planning for legged robots on varied terrain. *The Intl. Journal of Robotics Research*, 27(11–12):1325–1349, 2008.

A. Hereid and A. D. Ames. FROST: Fast robot optimization and simulation toolkit. *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 719–726, 2017a. ISSN 21530866. doi: 10.1109/IROS.2017.8202230.

A. Hereid and A. D. Ames. FROST: Fast robot optimization and simulation toolkit. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 719–726, 2017b.

D. Hershberger, D. Gossow, J. Faust, and W. Woodall. rviz – ROS Wiki. `http://wiki.ros.org/rviz`. [Accessed 23-Apr-2022].

J. Hodgins and M. H. Raibert. Biped gymnastics. *Dynamically Stable Legged Locomotion*, 79, 1988.

S. F. Hoerner. Fluid-dynamic drag. *Hoerner Fluid Dynamics*, 1965.

A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz. Anytime search-based footstep planning with suboptimality bounds. In *IEEE Intl. Conference on Humanoid Robots*, pages 674–679, 2012.

T. A. Howell, B. E. Jackson, and Z. Manchester. ALTRO: A fast solver for constrained trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7674–7679, 2019.

H.-M. Huang, K. Pavek, J. Albus, and E. Messina. Autonomy levels for unmanned systems (alfus) framework: An update. In *Unmanned Ground Vehicle Technology VII*, volume 5804, pages 439–448. International Society for Optics and Photonics, 2005.

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*, 35(12):1497–1521, 2016.

P. E. Hudson. *The structural and functional specialisation of locomotion in the cheetah (Acinonyx jubatus)*. PhD thesis, Royal Veterinary College (University of London), 2011.

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 *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 38–44, 2016.

M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch, et al. Anymal-toward legged robots for harsh environments. *Advanced Robotics*, 31(17):918–931, 2017.

IEEE Spectrum. Boston Dynamics' Cheetah Robot Now Faster than Fastest Human - IEEE Spectrum, 2012. URL `https://spectrum.ieee.org/automaton/robotics/military-robots/boston-dynamics-cheetah-robot-now-faster-than-fastest-human`.

S. International. Automated driving: levels of driving automation are defined in new sae international standard j3016, 2014.

J. M. Jimeno. GitHub - chvmp/champ: Quadruped robot based on MIT Cheetah I — github.com. `https://github.com/chvmp/champ`. [Accessed 16-Apr-2022].

A. M. Johnson and D. E. Koditschek. Legged self-manipulation. *IEEE Access*, 1:310–334, 2013a.

A. M. Johnson and D. E. Koditschek. Toward a vocabulary of legged leaping. In *IEEE Intl. Conference on Robotics and Automation*, pages 2568–2575, 2013b.

D. Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 1620–1626. IEEE, 2003.

M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal. Learning locomotion over rough terrain using terrain templates. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 167–172, 2009.

M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal. Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, 30(2):236–258, 2011.

M. Kapadia, A. Beacco, F. Garcia, V. Reddy, N. Pelechano, and N. I. Badler. Multi-domain real-time planning in dynamic environments. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 115–124, 2013.

S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE Conference on Decision and Control*, pages 7681–7687, 2010.

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The Intl. Journal of Robotics Research*, 30(7):846–894, 2011.

G. Kenneally, A. De, and D. Koditschek. Design Principles for a Family of Direct-Drive Legged Robots. *IEEE Robot. Autom. Lett.*, 1(2):900–907, 2016. ISSN 23773766. doi: 10.1109/LRA.2016.2528294.

D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.

S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann. RRT*-Connect: Faster, asymptotically optimal motion planning. In *IEEE Intl. Conference on Robotics and Biomimetics*, pages 1670–1677, 2015.

N. J. Kohut, A. O. Pullin, D. W. Haldane, D. Zarrouk, and R. S. Fearing. Precise dynamic turning of a 10 cm legged robot on a low friction surface using a tail. *IEEE Int. Conf. Robot. Autom.*, pages 3299–3306, 2013a. ISSN 10504729. doi: 10.1109/ICRA.2013.6631037.

N. J. Kohut, D. Zarrouk, K. C. Peterson, and R. S. Fearing. Aerodynamic steering of a 10 cm high-speed running robot. *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 5593–5599, 2013b. ISSN 21530858. doi: 10.1109/IROS.2013.6697167.

H. Kolvenbach, E. Hampp, P. Barton, R. Zenkl, and M. Hutter. Towards jumping locomotion for quadruped robots on the moon. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5459–5466. IEEE, 2019.

H. Kolvenbach, D. Wisth, R. Buchanan, G. Valsecchi, R. Grandia, M. Fallon, and M. Hutter. Towards autonomous inspection of concrete deterioration in sewers with legged robots. *Journal of field robotics*, 37(8):1314–1327, 2020.

M. Kovac, M. Fuchs, A. Guignard, J.-C. Zufferey, and D. Floreano. A miniature 7g jumping robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 373–378. IEEE, 2008.

J. J. Kuffner and S. M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *IEEE Intl. Conference on Robotics and Automation*, volume 2, pages 995–1001, 2000.

S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake. Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, 2016.

A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

T. Kunz and M. Stilman. Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 3713–3719, 2014.

V. Kurtz, R. R. da Silva, P. M. Wensing, and H. Lin. Formal connections between template and anchor models via approximate simulation. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 64–71. IEEE, 2019.

A. Lampe and R. Chatila. Performance measure for the evaluation of mobile robot autonomy. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 4057–4062. IEEE, 2006.

L. Landau and E. Lifschitz. *Fluid mechanics, Series in advanced physics, vol. 6*. Addison-Wesley, Reading, MA, 1959.

A. Laurenzi, E. M. Hoffman, and N. G. Tsagarakis. Quadrupedal walking motion and footstep placement through linear model predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2267–2273. IEEE, 2018.

S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *The Intl. Journal of Robotics Research*, 20(5):378–400, 2001.

J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

H. Li, R. Frei, and P. M. Wensing. Model hierarchy predictive control of robotic systems. *IEEE Robotics and Automation Letters*, 2021.

T. Libby, T. Y. Moore, E. Chang-Siu, D. Li, D. J. Cohen, A. Jusufi, and R. J. Full. Tail-assisted pitch control in lizards, robots and dinosaurs. *Nature*, 481(7380):181–184, 2012. ISSN 0028-0836. doi: 10.1038/nature10710.

T. Libby, A. M. Johnson, E. Chang-Siu, R. J. Full, and D. E. Koditschek. Comparative Design, Scaling, and Control of Appendages for Inertial Reorientation. *IEEE Trans. Robot.*, 32(6):1380–1398, 2016a. ISSN 15523098. doi: 10.1109/TRO.2016.2597316.

T. Libby, A. M. Johnson, E. Chang-Siu, R. J. Full, and D. E. Koditschek. Comparative design, scaling, and control of appendages for inertial reorientation. *IEEE Transactions on Robotics*, 32 (6):1380–1398, 2016b.

D. Limon, T. Alamo, D. M. Raimondo, D. Pena, J. M. Bravo, A. Ferramosca, and E. F. Camacho. Input-to-state stability: a unifying framework for robust model predictive control. In *Nonlinear model predictive control*, pages 1–26. Springer, 2009.

T. Litman. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.

G. H. Liu, H. Y. Lin, H. Y. Lin, S. T. Chen, and P. C. Lin. A bio-inspired hopping kangaroo robot with an active tail. *J. Bionic Eng.*, 11(4):541–555, 2014. ISSN 16726529. doi: 10.1016/S1672-6529(14)60066-4.

J. Liu, P. Zhao, Z. Gan, M. Johnson-Roberson, and R. Vasudevan. Leveraging the template and anchor framework for safe, online robotic gait design. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10869–10875. IEEE, 2020.

N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse. Using a memory of motion to

efficiently warm-start a nonlinear predictive controller. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2986–2993. IEEE, 2018.

J. H. Marden and L. R. Allen. Molecules, muscles, and machines: universal performance characteristics of motors. *Proceedings of the National Academy of Sciences*, 99(7):4161–4166, 2002.

E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *IEEE International Conference on Robotics and Automation*, 2010.

C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *IEEE International Conference on Robotics and Automation*, pages 2536–2542, 2020.

D. McConachie, T. Power, P. Mitrano, and D. Berenson. Learning when to trust a dynamics model for planning in reduced state spaces. *IEEE Robotics and Automation Letters*, 5(2):3540–3547, 2020.

R. B. McGhee and G. I. Iswandhi. Adaptive locomotion of a multilegged robot over rough terrain. *IEEE transactions on systems, man, and cybernetics*, 9(4):176–182, 1979.

K. Mombaur. Using optimization to create self-stable human-like running. *Robotica*, 27(3):321–330, 2009.

I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):1–8, 2012.

NASA. Home — Curiosity – NASA's Mars Exploration Program, 2020. URL `https://mars.nasa.gov/msl/home/`.

M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and

J. Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.

Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, and S. Kim. Optimized jumping on the mit cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454. IEEE, 2019.

J. Norby and A. M. Johnson. Fast global motion planning for dynamic legged robots. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3829–3836. IEEE, 2020.

J. Norby, J. Y. Li, C. Selby, A. Patel, and A. M. Johnson. Enabling dynamic behaviors with aerodynamic drag in lightweight tails. *IEEE Transactions on Robotics*, 37(4):1144–1153, 2021.

J. Norby, A. Tajbakhsh, and A. M. Johnson. Adaptive Complexity Model Predictive Control. 2022a. in prep.

J. Norby, Y. Yang, A. Tajbakhsh, J. Ren, J. K. Yim, A. Stutt, Q. Yu, and A. M. Johnson. Quad-SDK: Full stack software framework for agile quadrupedal locomotion. In *ICRA Workshop on Legged Robots*, May 2022b.

D. E. Orin, R. B. McGhee, and V. Jaswa. Interactive compute-control of a six-legged robot vehicle with optimization of stability, terrain adaptibility and energy. In *1976 IEEE Conference on Decision and Control including the 15th Symposium on Adaptive Processes*, pages 382–391. IEEE, 1976.

D. E. Orin, A. Goswami, and S.-H. Lee. Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35(2–3):161–176, 2013.

H.-W. Park, P. M. Wensing, S. Kim, et al. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In *Robotics: Science and Systems*, 2015.

A. Patel and M. Braae. Rapid turning at high-speed: Inspirations from the cheetah's tail. *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 5506–5511, 2013. ISSN 21530858. doi: 10.1109/IROS.2013.6697154.

A. Patel and M. Braae. Rapid acceleration and braking: Inspirations from the cheetah's tail. *IEEE Int. Conf. Robot. Autom.*, pages 793–799, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6906945.

A. Patel, E. Boje, C. Fisher, L. Louis, and E. Lane. Quasi-steady state aerodynamics of the cheetah tail. *Biol. Open*, 5(8):1072–1076, 2016. doi: 10.1242/bio.018457.

E. Plaku, L. E. Kavraki, and M. Y. Vardi. Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Transactions on Robotics*, 26(3):469–482, 2010.

R. Playter, M. Buehler, and M. Raibert. Bigdog. In *Unmanned Systems Technology VIII*, volume 6230, page 62302O. International Society for Optics and Photonics, 2006.

B. Ponton, M. Khadiv, A. Meduri, and L. Righetti. Efficient multicontact pattern generation with sequential convex approximations of the centroidal dynamics. *IEEE Transactions on Robotics*, 37(5):1661–1679, 2021.

M. Posa, C. Cantu, and R. Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The Intl. Journal of Robotics Research*, 33(1):69–81, 2014.

I. Poulakakis, J. A. Smith, and M. Buehler. On the dynamics of bounding and extensions: towards the half-bound and gallop gaits. In *Adaptive motion of animals and machines*, pages 79–88. Springer, 2006.

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

J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, 2017.

J. H. Reif. Complexity of the mover's problem and generalizations. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 421–427. IEEE Computer Society, 1979.

T. J. Roberts, E. M. Abbott, and E. Azizi. The weak link: do muscle properties determine locomotor performance in frogs? *Philosophical Transactions of the Royal Society B: Biological Sciences*, 366(1570):1488–1495, 2011.

Robohub. Packbot: Serving the Military and World Cup Football — Robohub, 2015. URL `https://robohub.org/packbot-serving-the-military-and-world-cup-football/`.

U. Saranli, M. Buehler, and D. E. Koditschek. Rhex: A simple and highly mobile hexapod robot. *The Intl. Journal of Robotics Research*, 20(7):616–631, 2001.

G. Schultz and K. Mombaur. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on mechatronics*, 15(5):783–792, 2009.

L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2641–2648. IEEE, 2006.

A. Shapiro, E. Rimon, and S. Shoval. A foothold selection algorithm for spider robot locomotion in planar tunnel environments. *Intl. Journal of Robotics Research*, 24(10):823–844, 2005.

J. M. Sheppard and W. B. Young. Agility literature review: Classifications, training and testing. *Journal of sports sciences*, 24(9):919–932, 2006.

Y. Shi, P. Wang, M. Li, X. Wang, Z. Jiang, and Z. Li. Model predictive control for motion planning of quadrupedal locomotion. In *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 87–92. IEEE, 2019.

K. Sreenath, H.-W. Park, and J. W. Grizzle. Design and experimental implementation of a compliant hybrid zero dynamics controller with active force control for running on mabel. In *2012 IEEE International Conference on Robotics and Automation*, pages 51–56. IEEE, 2012.

B. Styler and R. Simmons. Plan-time multi-model switching for motion planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pages 558–566, 2017.

H. S. Sushma and M. Singh. Hunting of Indian giant squirrel (Ratufa indica) by the lion-tailed macaque (Macaca silenus) in the Western Ghats, India. *Curr. Sci.*, 95(11):1535–1536, 2008. ISSN 00113891.

R. Tedrake. Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for mit 6.832. *Working draft edition*, 3, 2009.

R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL `https://drake.mit.edu`.

S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard. An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics*, 34(3):586–601, 2018.

M. Vukobratović and B. Borovac. Zero-moment point—thirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *New results and new trends in computer science*, pages 359–370. Springer, 1991.

M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter. Navigation planning for legged robots in challenging terrain. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 1184–1189, 2016.

A. M. Wilson, J. C. Lowe, K. Roskilly, P. E. Hudson, K. A. Golabek, and J. W. McNutt. Locomotion dynamics of hunting in wild cheetahs. *Nature*, 498(7453):185–189, 2013. ISSN 00280836. doi: 10.1038/nature12295.

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*, 3(3):1560–1567, 2018a.

A. W. Winkler, D. C. Bellicoso, M. Hutter, and J. Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3:1560–1567, July 2018b. doi: 10.1109/LRA.2018.2798285.

A. Witkin and M. Kass. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4):159–168, 1988.

G. J. Zeglin. Uniroo–a one legged dynamic hopping robot. B.S. Thesis, Massachusetts Institute of Technology, 1991.

H. Zhang, J. Butzke, and M. Likhachev. Combining global and local planning with guarantees on completeness. In *2012 IEEE International Conference on Robotics and Automation*, pages 4500–4506. IEEE, 2012.

J. Zhao, T. Zhao, N. Xi, M. W. Mutka, and L. Xiao. MSU Tailbot: Controlling Aerial Maneuver of a Miniature-Tailed Jumping Robot. *IEEE/ASME Trans. Mechatronics*, 20(6):2903–2914, 2015. ISSN 10834435. doi: 10.1109/TMECH.2015.2411513.

Y. Zhao and L. Sentis. A three dimensional foot placement planner for locomotion in very rough terrains. In *IEEE Intl. Conference on Humanoid Robots*, pages 726–733, 2012.

M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner. Optimization and learning for rough terrain legged locomotion. *The Intl. Journal of Robotics Research*, 30(2):175–191, 2011.