

Decoupled Kinodynamic Planning for a Quadruped Robot over Complex Terrain

by

Michael Burgess

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2023

© Michael Burgess, 2023. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Michael Burgess
Department of Mechanical Engineering
May 12, 2023

Certified by: Sangbae Kim
Professor
Thesis Supervisor

Accepted by: Kenneth Kamrin
Associate Professor
Undergraduate Officer

Decoupled Kinodynamic Planning for a Quadruped Robot over Complex Terrain

by

Michael Burgess

Submitted to the Department of Mechanical Engineering
on May 12, 2023, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Mechanical Engineering

Abstract

High-level planning for hybrid-dynamic, legged systems can be challenging due to a need to simultaneously satisfy kinematic and dynamic constraints. Previously developed sampling-based approaches can rapidly generate plans that satisfy kinematic constraints, but often lead to dynamically infeasible trajectories. On the other hand, traditional optimization-based approaches can reliably produce feasible trajectories, but are computationally inefficient. In this work, we leverage the strengths of these popular techniques to develop an advantageous novel motion planning formulation. Our methodology decouples kinematic and dynamic constraints to quickly generate emergent, feasible trajectories for legged systems across complex terrains. We decouple constraints into two separate processes. First, we rapidly sample footstep positions across a given terrain using an RRT-like search algorithm. This allows us to satisfy kinematic constraints without committing to a full state trajectory which could be dynamically infeasible, as is a common failure of other sampling-based approaches. Then, we can solve an optimization problem to generate a dynamically feasible trajectory using these contact positions. Since contact locations have already been determined, our optimization problem has a reduced decision space and does not require inconvenient complementarity constraints. As a result, this optimization can be solved more efficiently than traditional trajectory optimization formulations. Implemented in simulation for a 2D quadruped robot, our novel formulation is shown to generate trajectories in less than 15% of the computation time needed for traditional, coupled planning methods. Furthermore, experiments demonstrate that our method maintains a consistent average solve time across sets of randomly generated terrains, regardless of their complexity.

Thesis Supervisor: Sangbae Kim

Title: Professor

Acknowledgments

First and foremost, I would like to express my deepest gratitude to Matthew Chignoli, who was my direct mentor throughout this project. You have taught me so much about robotics, demonstrated great practices in writing and coding, and given me many other useful pieces of advice along the way. I am incredibly grateful for your kindness and support. Without your thoughtful guidance, I could not have completed this project.

Thank you to Prof. Sangbae Kim for supervising this project and empowering me as a member of his lab over these past years. Your wisdom has cultivated great success in the field of robotics. I cherish that I've had the privilege to learn from you and be included as part of The Biomimetic Robotics Lab.

I would like to recognize all of the teachers I had during and prior to this project for instilling confidence and intellectual curiosity in me. Perhaps unknowingly, you encourage me to continue in my pursuit of education everyday. I would like to particularly mention Mr. Ed Olson, Mr. Shawn Lampron, Ms. Gail Griffith, and Mr. Joel Penley.

My parents and sisters have provided me with limitless support throughout my life. Your love and pride mean the world to me. To my parents especially, you have exemplified to me what it means to work hard and why. Thank you for all that you have sacrificed to allow me to be where I am today. Also, I'd like to say congratulations to my wonderful older sister Sadie, and wish a lifetime of health and happiness to my newly born niece, Margo!

I would like to thank my girlfriend Michaela for sharing her love with me. You inspire me to be the very best version of myself in every way. I'd also like to thank her cat Chia for being super cute and the ultimate chiller.

Last but not least, I could not have made it through my undergraduate studies without my friends. They have been right there with me every day. Thank you each for finding your way into my life and filling all the otherwise mundane moments with laughter and joy. Your friendship is indispensable to me.

Contents

1	Introduction	15
1.1	Motivation	16
1.2	Related work	17
1.2.1	Optimization-based approaches	17
1.2.2	Sampling-based approaches	19
1.2.3	Contributions	20
2	Background	23
2.1	Rigid body dynamics	23
2.2	Rapidly-exploring Random Tree (RRT)	25
2.3	Trajectory optimization	27
3	Decoupled Planning	31
3.1	Framework	31
3.2	Contact sequence sampling	32
3.2.1	Kinematic feasibility	34
3.2.2	Connecting contact states	37
3.2.3	Sampling candidate contact states	37
3.2.4	Extracting a path	39
3.3	Dynamic optimization	41
3.3.1	Kinematic constraints	42
3.3.2	Dynamic constraints	44
3.3.3	Cost function	46
4	Results	47
4.1	Setup	47
4.1.1	Example trajectory generation	49
4.2	Benchmark against coupled approach	52

4.3	Performance analysis	55
5	Conclusion	57
5.1	Future work	58

List of Figures

1-1	A mountain goat stands atop a cliff, demonstrating its ability to navigate challenging terrain using its legs. The rugged and rocky landscape could not be traversed on wheels.	16
2-1	Full robot position state is described through linkage lengths and the angles between them. The Cartesian location of the center body position is described by (q_1, q_2) , with an angular pitch of q_3 . Elements of the robot state \mathbf{q} describe the position of the body and angles between linkages. The lengths of the front and rear leg linkages are equivalent.	24
2-2	An example image illustrating the process of an RRT search. Beginning at q_{start} , we sample new positions q_{rand} . During each iteration, we check whether this new position can be connected to any of its nearest neighbors q_{near} . We continue this process, and expand our undirected graph, until we reach the goal condition within tolerance of state q_{goal} .	25
3-1	Sampled contact positions for the front and rear limbs of the robot are highlighted across the terrain, denoted by $\mathbf{p}_{\text{front}}$ and \mathbf{p}_{rear} respectively. These positions are specified by Cartesian coordinates (x, z) . Additionally, the angle θ at which each limb makes contact with the terrain is specified.	33
3-2	To validate the sampled contact angle and prevent collision between the shank and terrain, the slope of the shank link is compared to the discrete terrain corners at position \mathbf{p}_{foot} along the terrain. The left sample is kinematically feasible because $ m_1 > m_{\text{corner}} $. The right sample is kinematically infeasible because $ m_2 < m_{\text{corner}} $	35

3-3	Possible body position angles for 2D quadruped given fixed knee positions. The space of possible body angles must expand from the edge of possible rear thigh angles. If the edges of the body and front thigh angle spaces intersect, it is guaranteed that the robot can maintain the knee positions chosen through sampling.	36
3-4	Centroidal free-body diagram for the robotic system. Contact forces $F_{x,\text{rear}}, F_{z,\text{rear}}$ are applied at \mathbf{p}_{rear} . And, contact forces $F_{x,\text{front}}, F_{z,\text{front}}$ are applied at $\mathbf{p}_{\text{front}}$. Gravity acts in the $-\hat{z}$ direction from the centroidal position (c_x, c_z)	45
4-1	Example of a generated terrain. The terrain has a fixed length L in the \hat{x} direction. The initial, middle, and final sections have a constant height of 0 in the \hat{z} direction. The 2 nd and 4 th sections are designated as obstacles. These sections have randomized heights and widths within a specified possible range.	48
4-2	A sequence of robotic poses over a generated terrain with obstacle heights of 0.2m and 0.1m. Foot and knee positions are found through our RRT-based sampling algorithm. The corresponding full robotic states are then obtained through inverse kinematics. To reduce clutter in the image, only a few sampled poses from the discrete trajectory are displayed. The color of plotted robot positions change from red to blue as they move from the initial to the final state.	50
4-3	The planned centroidal trajectory of the robot's body is plotted over a generated terrain at both stages of the planning process. Discrete centroidal body positions are extracted from sampled contact positions through inverse kinematics. These positions are then used in cost function of dynamic optimization. The continuous, optimized trajectory is displayed in relation to these sampled points.	51
4-4	The angle about the center of the robot's main body linkage is shown over the duration of the same trajectory shown in previous graphs. This pitch angle ϕ is shown to be nearly zero across the trajectory. . .	52

4-5	The trajectory obtained from the benchmark coupled optimization is plotted over a step terrain that increases in height from 0 to 0.2 meters. The terrain is fully described by (4.1). The continuous centroidal trajectory is displayed along with a selected subset of full robot poses sampled along the trajectory. The robot positions transition from red to blue as they move from the initial to the final state.	53
4-6	Average solve times are presented with standard error bounds for the coupled and decoupled approaches over 100 trials on both flat and non-flat terrains. The length of each terrain is 1.2m, and the non-flat terrain features approximates a discrete step up as defined by (3.4). .	54
4-7	Our decoupled implementation was ran over a set of terrains with varying degrees of complexity. The average time needed to find a path across each of these generated terrain sets is plotted with standard error bounds. Bars are divided based on the time consumed by each sub-process involved in the planning approach, namely RRT-based contact sampling and dynamic optimization. The success rate of our overall implementation across these terrains is plotted as well. All generated terrains had a constant total length of 2.25m.	56

List of Tables

4.1	Robot System Parameters	48
4.2	Cost Function Weights	49
4.3	Terrain Difficulty	55

Chapter 1

Introduction

Humans and animals can smoothly traverse complex environments, while rarely pausing to make a decision. We seek to replicate this behavior in robots. To achieve complete autonomy, robots must be able to quickly generate practical paths across all kinds of terrain, just as an animal would. If we could accomplish this task in real-time, a robot would be capable of navigating throughout the world on its own and intelligently reacting to dynamic environment changes.

We aim to develop better planning frameworks that could enable the fast decision-making we observe in animals. While extensive research has been dedicated to developing robust planning systems for robot locomotion, high-level planning frameworks remain computationally expensive over challenging terrain. Solving these kinodynamic planning problems requires us to consider the kinematics and dynamics of the system simultaneously. These two factors are deeply intertwined and perpetuate the difficulty of the problem.

This thesis seeks to provide a new methodology for planning high-level robotic motion that offers advantages over the current state-of-the-art solutions. This is done in an effort to move toward faster planning frameworks. Most crucially, we present a new approach that decouples kinematic and dynamic constraints on the legged planning problem. For the purposes of this work, we will limit our scope and focus exclusively on high-level planning for legged locomotion, specifically walking over non-flat terrain. This mode of locomotion is necessary for traversing complex, discrete environments, but it poses challenges in planning because it is inherently hybrid-dynamic.

1.1 Motivation

Autonomous mobile robots have the potential to greatly benefit our society through a variety of applications. One of the most significant advantages they could provide is their capacity to perform monotonous or arduous tasks that people may find unpleasant. But, more importantly, robots could operate in hazardous environments that are unsafe for humans. In particular, legged robots offer increased ability to navigate rugged, and potentially dangerous, terrain. We see legged animals, like the mountain goat in Figure 1-1, demonstrate this ability all the time. Unlike wheeled robots, legged robots would not be limited to smooth landscapes, making them especially useful in emergency situations. Legged robots could be designed to traverse and perform tasks on unstable buildings, challenging elevated locations with inadequate footing, as well as radioactive or chemically dangerous areas.



Figure 1-1: A mountain goat stands atop a cliff, demonstrating its ability to navigate challenging terrain using its legs. The rugged and rocky landscape could not be traversed on wheels.

Further developments in robotic navigation are needed to enable these sorts of applications. Legged robots require more sophisticated planning architectures than simple wheeled robots. Legged systems have more degrees of freedom, but they must also consider the hybrid-dynamics of making and breaking contact with the ground. Discrete decisions like contact placement can be challenging for planning frameworks that must also consider the continuous dynamics of the system. The computational tools suited for these distinct types of decision making often conflict. This is an inherent problem with legged planning, but the advantage of these systems

is that discrete footsteps allow for the traversal of rough terrain. Planning frameworks must overcome this challenge and be capable of rapidly generating hybrid-dynamic trajectories. This is essential for an autonomous legged robot to respond promptly to unforeseen events or dynamic changes in its surroundings.

1.2 Related work

Most high-level planning frameworks fit within two categories: sampling-based and trajectory optimization-based. This section provides brief overviews of existing approaches in both of these categories, and details their relative advantages and shortcomings.

1.2.1 Optimization-based approaches

Optimization presents a great way to solve problems with continuous variables subject to constraints, especially when these constraints are complex [2]. Optimization can be utilized to generate reliable high-level robotic plans. We can develop a model of our system’s dynamics and solve for a trajectory that is constrained to obey these dynamics [11]. However, for a robotic system with numerous degrees of freedom, like a quadruped, the decision space of these optimizations problems can grow quite large. This means that solve times are relatively inefficient. Furthermore, since kinodynamic trajectory optimization problems tend to be nonlinear, generated solutions are not guaranteed to be optimal.

Optimization can be leveraged to handle discrete events like contact using complementarity constraints [37, 36]. These constraints are necessary for optimizing a hybrid trajectory such as walking, but are problematic for solvers. They have discontinuous gradients and violate the linear independence constraint qualification assumed by conventional solvers [34]. This fact, in combination with the large decision space of walking problems, significantly restrains the efficiency of optimization-based approaches to high-level planning.

The computational efficiency of optimization-based approaches can be improved by considering a reduced order model of the robot. This strategy restricts the decision space of the problem to decrease solving time. One way to achieve this is by only considering dynamics about the robot’s center body position [9]. Although reduced order models neglect some of the robot’s dynamics, simplified trajectories can still be successfully executed by implementing a stabilizing whole-body controller [27, 24].

Still, our reduced model is less accurate, so resulting trajectories are not guaranteed to be trackable, even with a stabilizing controller. This loss of accuracy is a direct tradeoff for increased solving efficiency. Even with partially improved efficiency, kinematic contact constraints must still be imposed on the problem. These constraints must consider the robot’s full body state and are often nonlinear, making the problem non-convex [14]. Consequently, solutions are not guaranteed to be optimal and may take longer to reach [4].

Mixed-integer approaches to optimization have been used to make the footstep planning problem convex. By relaxing nonlinear kinematic constraints, this methodology can be used to guarantee an optimal solution is found for the approximated problem [10, 26]. This technique requires creating more constraints that are switched on and off. Thus, in order to solve the problem, many optimizations must be solved along the combinatorial decision tree of possible constraint sets. Despite the guarantee of an optimal trajectory, this drastically increases the relative time needed to solve for that trajectory. Similarly, a mixed-integer style approach can be implemented to relax the centroidal dynamics of a robot, making the whole planning problem convex [35]. Nonetheless, this demonstrates the same shortcomings associated with other mixed-integer planning methods.

Many optimization-based approaches to legged planning rely on a pre-scheduled gait. This helps reduce the decision space of the problem by constraining when each foot should be in contact with the ground. However, efforts have been made to eliminate this assumption and create more general forms of trajectory optimization that generate emergent gaits. For instance, optimizations have been formulated where decision variables form splines of stepping motions [42]. This approach gives an expressive path with implicit gait. Switched system optimal control approaches have also been employed to solve for emergent gaits by changing control with contact [15]. Additionally, new methods of contact-invariant optimization can be used to smooth discrete contact events into continuous trajectories with emergent gaits [32].

Model-based predictive control (MPC) can be used to control complex robotic systems with nonlinear dynamics and constraints. This approach works by predicting the future behavior of a system through a model and determining the optimal control input to achieve a desired performance. However, it is computationally intensive, requiring repeated online optimization, and it relies heavily on the accuracy of the system model used for prediction [33, 18, 6, 11].

In general, trajectory optimization based approaches to robotic path planning trade off slow solution time for dynamic feasibility. Hierarchical frameworks are used

to mitigate the limitations presented by this [43, 30, 12, 19]. Under a hierarchical architecture, long-term plans can be generated at a lower frequency with complex modeling, while short-term plans are generated at a high frequency with reduced modeling. Similar frameworks have been developed to generate low-fidelity, geometric paths that can later be refined into high-fidelity, dynamically feasible paths [7]. These are strong methodologies to create operative planning frameworks given the limitations of optimization-based approaches.

1.2.2 Sampling-based approaches

Sampling-based approaches are commonly used to efficiently generate high-level plans for legged robotic systems. One such methodology is the Rapidly-exploring Random Tree (RRT) algorithm, which has shown to be effective at sampling paths in a variety of problem spaces [25, 29, 21, 13]. RRT is a probabilistically complete search algorithm. So, it is guaranteed to find a path if one exists, but it is not guaranteed to find an optimal path [40]. In our application, this is admissible because we only require a feasible traversal path, not necessarily an optimal one. RRT gives the benefit of making discrete decisions over a large, high-dimensional space of possible robotic states at a computationally efficient speed. Moreover, it allows for free and expressive movement through the environment. It does not restrict the robot’s path to some prescribed gait, since all chosen states are randomly sampled. Poses generated from this type of search are typically fed into an optimization problem to stabilize them into a continuous, feasible trajectory.

Although sampling-based methods have the benefit of efficiency, they face difficulties in reliably producing feasible plans for hybrid-dynamic systems, such as walking robots. The search process often yields pose sequences that lead to dynamically infeasible trajectories. A crucial part of RRT is determining the nearest neighbors of a state [8, 25]. In a planning application, these neighbors are analyzed to determine connectable states which the robot could move between. Traditionally, neighboring states are found using a simple Euclidean distance metric. However, this is not the most accurate way to calculate the ability to move between full robotic position states. Thus, it is difficult to create contact-rich, dynamic maneuvers through this type of approach [31, 5].

Variants of RRT have been developed in attempt to improve the effectiveness of the approach for kinodynamic systems. Instead of determining neighbors using simple Euclidean distance, reachability guided methods (RG-RRT) consider the feasibility of

transitioning between neighboring states to enable a more accurate and effective nearest neighbors query [38, 39]. This yields a stronger framework for planning dynamic motions. This type of approach has been extended into an alternative form called environment-guided RRT (EG-RRT) that biases towards promising parts of the state space [20]. And, a more formal method has been developed in [44]. Despite the improvement shown by these methods, they do not solve the problem. They attempt to bias sampling towards more states that are more likely dynamically feasible [3]. But, this bias towards certain states is not a strict constraint. As a result, these methods are not very effective at planning for highly dynamic legged systems in practice.

Alternative sampling-based approaches have been explored to create acyclic contact plans that approximately satisfy static equilibrium by checking a reachability condition [41]. This criterion can be used to assess the practicality of a potential state and facilitate more effective sampling. While this method does not directly utilize RRT, it still experiences similar pitfalls. It is susceptible to producing dynamically infeasible pose sequences because its constraints are solely kinematic.

As demonstrated by the above summarized works, sampling-based approaches give the advantage of computation speed but do not reliably lead to dynamically feasible solutions. Furthermore, they are predicated on random sampling and have no notion of optimality or cost. Meanwhile, optimization-based approaches allow us to easily quantify and compare trajectories through a cost function, sampling-based approaches do not possess this feature.

1.2.3 Contributions

Both classes of approaches provide their own characteristic successes and failures. Sampling-based techniques excel in handling discrete planning decisions. They are typically more computationally efficient, but scale poorly as constraints are added. Optimization-based methods are better suited to make continuous decisions. They produce reliably feasible trajectories, but are often time-consuming to execute. We would like to explore how to strike a balance between these methodologies. The remainder of this thesis is dedicated investigating how to draw from the relative advantages of both approaches to develop a new planning methodology.

In this thesis, a novel approach to path planning for hybrid-dynamic, legged systems will be presented. This approach effectively decouples the kinematic and dynamic constraints of a robotic system in attempt to rapidly produce feasible trajectories across complex terrains. Sampling-based methods will be used to quickly

search for kinematically feasible contact positions. The resulting sequence of contact positions can be fed into a dynamic optimization to find a dynamically feasible centroidal trajectory. This optimization problem can be solved quicker than traditional optimization methods because foot contact positions have already been found, so the decision space of the planning problem is greatly reduced and problematic complementarity constraints are not needed. Overall, this novel implementation facilitates the rapid calculation of dynamically viable trajectories, obtaining solutions in less than 15% of the time required by comparable popular implementations.

Chapter 2

Background

Traditional forms of high-level robotic planning often couple kinematic and dynamic constraints into one planning problem. This is done in attempt to find a trajectory through an environment that is both kinematically and dynamically feasible. There are a number of approaches to this problem. The two most popular are sampling-based methods and direct trajectory optimization. This chapter aims to provide background information on these common approaches, which is necessary to understand the novel concepts presented later in the thesis.

2.1 Rigid body dynamics

Rigid body dynamics is a fundamental concept in robotics that plays a critical role in the design and control of legged robots. Rigid bodies are defined as objects that maintain their shape and size regardless of external forces acting on them. In the context of legged robots, rigid body dynamics deals with the motion of the robot's body and limbs, and how they interact with the environment. Understanding the principles of rigid body dynamics is essential for developing control algorithms that enable legged robots to perform complex tasks such as walking, running, and jumping. In this section, we will provide a brief introduction to rigid body dynamics and its relevance to legged robot motion planning, and present the dynamic model for our quadruped robot.

In the case of our 2D quadruped robot, the system can be abstracted to a set of 5 linkages. There is one central body linkage, a front thigh link, front shank link, rear thigh link, and rear shank link. Legs are comprised of thigh and shank linkages. The rear leg is identical to the front leg. On both legs, hip and knee angles are directly actuated. The pose of the body is regulated through this leg actuation and

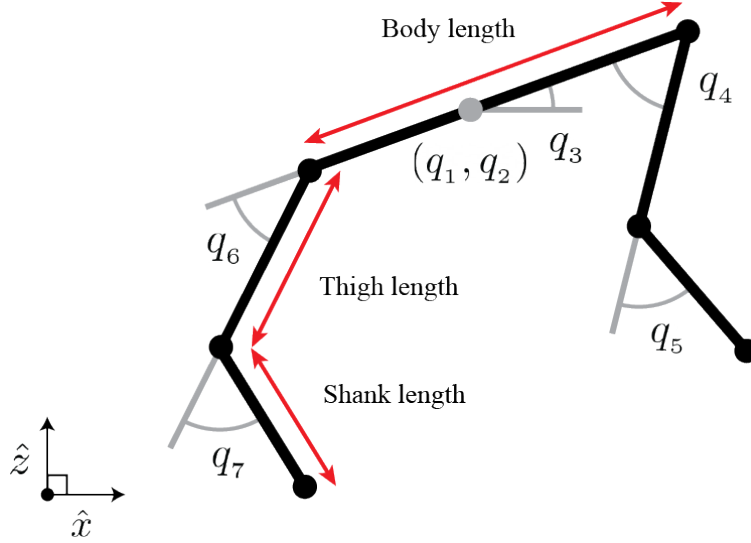


Figure 2-1: Full robot position state is described through linkage lengths and the angles between them. The Cartesian location of the center body position is described by (q_1, q_2) , with an angular pitch of q_3 . Elements of the robot state \mathbf{q} describe the position of the body and angles between linkages. The lengths of the front and rear leg linkages are equivalent.

consequent interaction with the ground. To plan the motion of the robot, we find a trajectory for the centroidal body position and each of these joint angles. These values are expressed as the full state of the robot \mathbf{q} . This state is defined by (2.1) and the corresponding robot geometry is illustrated in Figure 2-1.

$$\mathbf{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 \end{bmatrix} \quad (2.1)$$

Equations of motion for the robot can be derived through Lagrangian Mechanics and take on the form shown in (2.2). This equation can be used to model the behavior of every rigid link in the robot body.

$$\mathbf{H}\ddot{\mathbf{q}} + \mathbf{C} = \boldsymbol{\tau} + \mathbf{J}^T \mathbf{F} \quad (2.2)$$

To solve a planning problem, we compute the forward dynamics of the system through our equation of motion. This entails finding the acceleration of the system $\ddot{\mathbf{q}}$ from a given state \mathbf{q} and joint torques $\boldsymbol{\tau}$. Joint torque $\boldsymbol{\tau}$ is our control input. The forward dynamics tell us how this input impacts the motion of our robot, where output is the robot's state \mathbf{q} and its derivatives. The forward dynamic computation is executed repeatedly in attempt to satisfy constraints and choose optimal torque $\boldsymbol{\tau}$ inputs that facilitate desired behavior of our system.

To simplify system dynamics, we can project the dynamics of each linkage onto the center position of the system. Then, we only need one equation of motion for the entire system, instead of independent dynamic equations for each linkage. This reduced order model is called the centroidal dynamics of the system. Our now simplified differential equation can be efficiently solved using recursive algorithms documented in [16].

2.2 Rapidly-exploring Random Tree (RRT)

Rapidly-exploring Random Tree (RRT) is a randomized search algorithm that can be used to efficiently explore paths across a high-dimensional state space [28]. Beginning at a starting node q_{start} , new nodes q_{rand} are randomly sampled in the state space and added to a graph of visited states. The nearest neighbors q_{near} to this new node q_{rand} can be computed over the existing graph [8]. If there is a feasible connection between these neighboring states and the new node, the pair can be connected by an edge in the graph. This is done repeatedly until the new node is connected and satisfies some goal condition. Once complete, we have a graph G of connected states through our environment. From this graph, we can trace a path from our final node which satisfies the goal condition back to our initial node q_{start} . A diagram of the graph of states is provided in Figure 2-2. Pseudocode for the algorithm is provided in Algorithm 1.

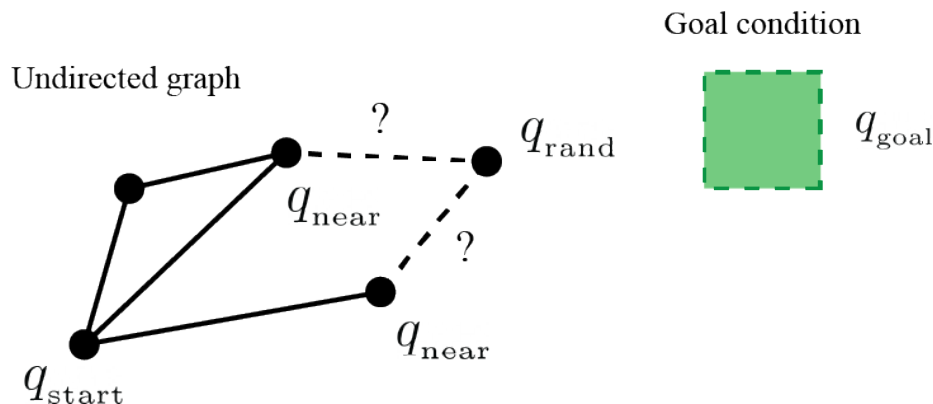


Figure 2-2: An example image illustrating the process of an RRT search. Beginning at q_{start} , we sample new positions q_{rand} . During each iteration, we check whether this new position can be connected to any of its nearest neighbors q_{near} . We continue this process, and expand our undirected graph, until we reach the goal condition within tolerance of state q_{goal} .

RRT is a probabilistically complete search algorithm [40]. This means that it is guaranteed to find a path between designated initial and final states if one exists.

Algorithm 1: Rapidly-Exploring Random Tree (RRT)

Input: Starting state q_{start} , Goal state q_{goal} , Maximum iterations N_{max}
Output: Graph of connected states G

- 1 Initialize tree with root: $G \leftarrow q_{\text{start}}$
- 2 **for** $i \leftarrow 1$ **to** N_{max} **do**
- 3 $q_{\text{rand}} \leftarrow$ Randomly sample configuration
- 4 $q_{\text{near}} \leftarrow$ NEAREST_NEIGHBOR(q_{rand}, G)
- 5 $q_{\text{new}} \leftarrow$ EXTEND($q_{\text{near}}, q_{\text{new}}, \Delta q$)
- 6 **if** q_{new} is collision-free **then**
- 7 $G.\text{add_vertex}(q_{\text{new}})$
- 8 $G.\text{add_edge}(q_{\text{new}})$
- 9 **if** q_{new} is sufficiently close to q_{goal} **then**
- 10 **return** G
- 11 **end if**
- 12 **end if**
- 13 **end for**

However, the optimality of a path found cannot be guaranteed because the search method is sampling-based and randomized. We do not have a cost function to quantify optimality or bias certain regions of the state space. Sampled states are only constrained to be collision-free. Additionally, the algorithm cannot confirm whether a path between the two points exists or not; it simply continues searching until one is found or until a maximum iteration limit N_{max} is reached. If the goal condition cannot be satisfied in less than N_{max} iterations, we should prematurely terminate the algorithm and assume a path cannot be found.

Although the RRT algorithm is not guaranteed to find an optimal solution, it is a widely used and efficient method for path planning in high-dimensional spaces. In a robotic planning application, each node in the graph would be a state of the robot. The initial node would be where the robot is. And, the final node would be where we want the robot to be. Robotic motion planning is an innately a high-dimensional search problem. Robots have large state spaces and the environments they are meant to explore can be quite vast and cluttered with obstacles. Therefore, RRT is great way to find a possible path between two robot positions due to its incredible computational efficiency.

Furthermore, the aforementioned drawbacks of RRT are inconsequential in most robotic planning applications. If we're planning motion in a vast environment, we can assume that a feasible path is possible, so we don't have to worry about the algorithm getting stuck. Plus, we don't necessarily care if a path through in our environment is

optimal or not, we just want to make sure the robot can feasibly move from one point to another. For these reasons, RRT is a strong approach to the problem of robotic planning.

2.3 Trajectory optimization

Optimization is a great way to find a solution to a continuous problem subject to some constraints [2]. The problem requires a set of decision variables, input parameters, constraints, and a cost function. A very simple example optimization formulation is provided in (2.3). In optimization, decision variables are what we hope to determine the value of. Here, our decision variables comprise some vector \mathbf{x} . The relative optimality of these variables is determined by a cost function $J(\mathbf{x})$. We would like to find the values of our decision variables to minimize this cost function. Input parameters can also be used in our optimization. Parameters are known variables that can be used to construct constraints on what values the decision variables are allowed to maintain. Constraints are functions of decision variables and parameters. Overall, we minimize our cost function subject to these constraints.

$$\begin{aligned} \min_{\mathbf{x}} \quad & J(\mathbf{x}) \\ \text{s.t.} \quad & f(\mathbf{x}) = 0 \\ & g(\mathbf{x}) \leq 0 \end{aligned} \tag{2.3}$$

Trajectory optimization refers to the process of finding a trajectory or path for a robot to follow through the use of a carefully formulated optimization problem. In robotics, we define a state \mathbf{q}_i that represents a configuration of our robot in the environment. Planning requires finding a feasible sequence of states for our robot. In the real world, this sequence would be a continuous trajectory. However, to formulate an optimization problem, we must break this continuity into a finite number of discrete points N . This process is called transcription. Each discrete point will have a timestep between them of duration h . We can then construct our discrete trajectory as a vector of states \mathbf{q} with length N , following (2.4). A continuous trajectory can later be extracted as the spline between these points.

$$\mathbf{q} = \left[\mathbf{q}_0 \quad \mathbf{q}_1 \quad \mathbf{q}_2 \quad \dots \quad \mathbf{q}_{N-1} \right] \tag{2.4}$$

At a given state \mathbf{q}_i , we have the decision to take an action \mathbf{u}_i . This action is a

member of our control space \mathcal{U} of possible actions, as given by (2.5).

$$\mathbf{u}_i \in \mathcal{U} \tag{2.5}$$

We require a discrete trajectory of control inputs, or actions, to coincide with our trajectory of states \mathbf{q} . This trajectory \mathbf{u} should be of the same length N , as defined in (2.6).

$$\mathbf{u} = \left[\mathbf{u}_0 \quad \mathbf{u}_1 \quad \mathbf{u}_2 \quad \dots \quad \mathbf{u}_{N-1} \right] \tag{2.6}$$

Actions and states are deeply intertwined. A chosen action \mathbf{u}_i will affect future states. Conversely, a state \mathbf{q}_k is a function of previous actions \mathbf{u}_i where $i \in (0, k - 1)$ and the robot's initial configuration \mathbf{q}_0 . Considering these facts, there are a couple of ways we can approach formulating our optimization problem. We know that we need to find a sequence of states \mathbf{q} and control inputs \mathbf{u} . A "direct collocation" approach uses both trajectories \mathbf{q} and \mathbf{u} decision variables of the optimization [23]. We would search potential values of both of these vectors in attempt to minimize our cost function. In this thesis, the chosen optimization form is direct collocation.

Another approach, called "direct shooting", uses only control inputs \mathbf{u} as decision variables. Here, the state trajectory \mathbf{q} is determined in response to control inputs and the robot's initial configuration \mathbf{q}_0 [23]. This approach will not be used. It obfuscates the ability to impose simple kinematic constraints on the path of our robot, since intermediate robot states are no longer part of our decision variables [2].

With our decision variables chosen, we can set a minimal list of constraints and form an optimization problem. First, we should bound the decision variables with upper and lower limits. Additionally, we must constrain the system to follow its dynamics. This can be formulated using a function f such that $\dot{\mathbf{q}}_i = f(\mathbf{q}_i, \mathbf{u}_i)$. This tells us that the motion of our system at point i depends on our state \mathbf{q}_i and what action we take at that state \mathbf{u}_i . We can use Euler integration to constrain future states based on the timestep Δt between indices i and $i + 1$. By converging on a solution to this problem, we find a set of decision variables that satisfies our list of constraints. This should also satisfy a local or absolute minima of our cost function

J . Cost function and constraints are fully formulated into (2.7).

$$\begin{aligned}
\min_{\mathbf{q}, \mathbf{u}} \quad & J(\mathbf{q}, \mathbf{u}) \\
\text{s.t.} \quad & \mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max} \\
& \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \\
& \dot{\mathbf{q}} = f(\mathbf{q}, \mathbf{u}) \\
& \mathbf{q}_{i+1} = \mathbf{q}_i + \dot{\mathbf{q}}_i \Delta t
\end{aligned} \tag{2.7}$$

More constraints could be added to the problem formulation. Constraints can be functions of control inputs, states, or any other input parameters. Moreover, they can be constructed to satisfy inequalities or equalities. Inequality constraints g_i and equality constraints g_e should maintain the same form as (2.8). The exact constraints we choose to add depends on the problem we are trying to solve. In path planning, these could be terrain dependent or more complex.

$$\begin{aligned}
g_i(\mathbf{q}_i, \mathbf{u}_i) &\leq 0 \\
g_e(\mathbf{q}_i, \mathbf{u}_i) &= 0
\end{aligned} \tag{2.8}$$

We have developed a cost function J , but we are not guaranteed to find optimal \mathbf{q} and \mathbf{u} that satisfy the global minimum of this function. Optimization problems can be either convex or non-convex. A convex optimization problem has a single optimal solution with absolute minimum cost J that can be easily found using a gradient-based solver [4]. On the other hand, a non-convex problem will have many local optima. Thus, a gradient-based solver may be prone to getting stuck these local optima and return sub-optimal solutions. Convexity depends on imposed constraints and the chosen cost function. If constraints are nonlinear in the decision variables, the problem becomes non-convex. Similarly, if the cost function is not quadratic, the problem may become non-convex. So, if we impose only linear constraints and make our cost function quadratic, we can guarantee an optimal solution is found. Typically, kinodynamic problems of robotic path planning require nonlinear kinematic constraints. Therefore, corresponding optimization problems are non-convex. We must solve these problems using some nonlinear solver like IPOPT or SNOPT [45, 17].

Trajectory optimization is great for motion planning because it allows us to encode the dynamics of a complex robot into the planning process. Furthermore, it allows to search for an optimal path according to our cost function. Sampling-based methods don't have this notion of optimality because they do not have a cost function.

When applied to a legged system at high-level, our state \mathbf{q}_i is as described by (2.1) at each index i . So, we get a trajectory of robot states at every timestep. To enable planning, we likely need to constrain our first state to be at some starting position, and the final state to be around some final goal position. Our control variable \mathbf{u}_i could be the force of contact between the robot and the ground at discrete point i . These forces are bounded based on the capabilities of our robot's actuators. Combining states and actions, we can get a full trajectory of positions and necessary ground reaction forces to move the robot across some environment from its initial state to some goal.

Chapter 3

Decoupled Planning

Trajectory optimization is often used to generate dynamically feasible trajectories, but this approach can be time-consuming when applied to complex problems like planning for legged systems. Alternatively, sampling-based approaches can efficiently find a full sequence of robotic poses, but the resulting path may be dynamically infeasible. To strike a better balance between these two approaches, we aim to investigate a method that leverages the strengths of each, rather than relying heavily on one technique alone. By developing a hybrid approach that combines the benefits of both trajectory optimization and sampling-based methods, we aim to efficiently generate solutions that are dynamically feasible, avoiding the limitations that each method experiences in isolation.

3.1 Framework

A novel framework will be developed to investigate integration between rigorous dynamic optimization and efficient kinematic sampling. A summary of this framework is described in this section. Further details are explained in later sections of this chapter.

To start, we sample foot contact positions for our robot across a terrain. This enables us to generate a sequence of feasible footstep positions that satisfy kinematic constraints, such as enforcing no collisions. However, it does not confine us to a set of full states that may be dynamically infeasible, as is a common failure of other sampling-based approaches to planning. Full state positions can be extracted from these contact positions through inverse kinematics. Then, the resulting sequence of poses can be used to help fit a centroidal trajectory through a dynamics-based trajectory optimization. This optimization will find a dynamically feasible path for

the robot given the sampled contact positions. Since we have already chosen our foot contact positions through sampling, this optimization problem has a reduced decision space and does not require nonlinear complementarity constraints. This leads to increased solving efficiency.

Through this approach, kinodynamic constraints are effectively decoupled. The sampling process handles kinematic constraints on the terrain and robot system. Once these are satisfied, the optimization problem only has to worry about the dynamics of the system. These processes work with each other to efficiently find a high quality path across the given terrain. Each works to account for the lapses that are characteristic to the other approach. Dynamic optimization ensures that sampled positions can be turned into a feasible path. It is granted the freedom to do so because we are not sampling full robotic states. We only sample foot positions, and have degrees of freedom to choose the corresponding body positions. Moreover, by sampling contact positions, we reduce the number of decision variables and constraints needed for the optimization, thereby decreasing the time needed for the solver to find a feasible solution. Since we have already determined a gait and exact footstep locations through our sampling algorithm, we only need to focus on contact forces and centroidal positioning in our optimization.

In the following, a decoupled planning approach to high-level planning is detailed for a 2D abstraction of quadruped. Our approach is not confined to this system alone. An analogous approach could be developed for any legged robotic system.

3.2 Contact sequence sampling

We would like to rapidly generate a sequence of kinematically feasible poses \mathbf{q} for our robot across a given terrain. This sequence can later be used to optimize a dynamically feasible trajectory. We start by finding the places where the robot should make contact with the ground. In other words, we find a sequence of footstep positions for the robot \mathbf{p} . This is an advantageous approach because it allows us to satisfy kinematic constraints without restricting ourselves to a full state trajectory that may be dynamically infeasible.

In this implementation, we are planning for a 2D quadruped system. This system effectively has two feet. To fully describe how the robot makes contact with the ground, we must sample the position of both the front foot $\mathbf{p}_{\text{front},i}$ and rear foot $\mathbf{p}_{\text{rear},i}$, as they are defined in (3.1). The full contact state \mathbf{p}_i of the robot concatenates these vectors, following (3.2). In a general implementation, the size of \mathbf{p}_i would grow

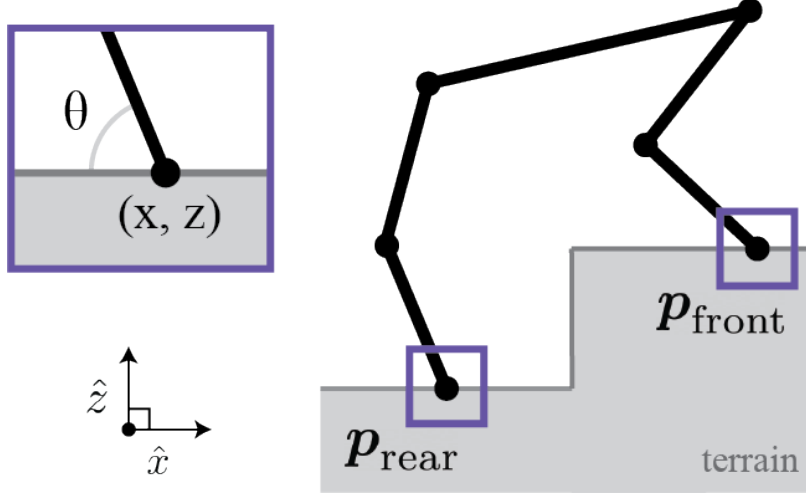


Figure 3-1: Sampled contact positions for the front and rear limbs of the robot are highlighted across the terrain, denoted by $\mathbf{p}_{\text{front}}$ and \mathbf{p}_{rear} respectively. These positions are specified by Cartesian coordinates (x, z) . Additionally, the angle θ at which each limb makes contact with the terrain is specified.

linearly with number of feet. Contact positions should include the Cartesian location of the foot and the angle which the shank linkage makes contact with the ground θ . This gives us a full picture of how the robot interacts with the ground. Contact positions relative to the full 2D robot are displayed in Figure 3-1.

$$\mathbf{p}_{\text{front},i} = \begin{bmatrix} x_{\text{front},i} \\ z_{\text{front},i} \\ \theta_{\text{front},i} \end{bmatrix}, \quad \mathbf{p}_{\text{rear},i} = \begin{bmatrix} x_{\text{rear},i} \\ z_{\text{rear},i} \\ \theta_{\text{rear},i} \end{bmatrix} \quad (3.1)$$

$$\mathbf{p}_i = [\mathbf{p}_{\text{front},i} \quad \mathbf{p}_{\text{rear},i}] \quad (3.2)$$

We use an RRT-based search algorithm to sample contact positions across the terrain. Beginning at some starting set of contact positions \mathbf{p}_0 , we randomly sample new contact positions \mathbf{p}_i until a goal position is reached. For a given terrain, this goal condition lies at the end of the terrain, opposite to the starting position of the robot. Throughout the sampling process, we construct an undirected graph G of feasible contact states. Edges on this graph represent movement between contact positions. When sampling is complete, this connected graph enables us to extract a connected path of contact poses \mathbf{p} that allow us to traverse the terrain from start to finish. The sequence is outlined by (3.3). Since footstep positions are randomly sampled, this

sequence defines a gait with emergent footstep order, locations, and overall pattern.

$$\mathbf{p} = \left[\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_2 \quad \dots \quad \mathbf{p}_{N_{\text{sampled}}} \right] \quad (3.3)$$

3.2.1 Kinematic feasibility

Primarily, we want to make sure that all of the contact states we sample are kinematically feasible. That way, we can limit the number of kinematic constraints in our optimization problem. For a robot pose to be considered kinematically feasible, a few conditions must be met:

1. Contact poses must not collide with the terrain.
2. The robot must be able to configure itself with feet at sampled contact positions.
3. The robot must be able to move from one contact position to the next.

Beginning with randomly sampled positions for each robot foot, we evaluate a series of ordered conditions based on the above statements to examine whether or not foot positions enable a kinematically feasible robot state. If any of these checks fail, the state is deemed infeasible and we go back to sampling. The process of sampling new contact states is documented in section 3.2.3. Every kinematic conditional checks is designed to be computed efficiently. They are ordered with decreasing efficiency to make sure infeasible states can be dispelled as fast as possible. The process of ensuring a new state is kinematically feasible for our implementation is documented below.

First, we can easily make sure that sampled foot positions do not collide with the terrain. We directly sample contact position x_i along the 2D terrain. Given a map of the terrain, we can determine the corresponding ground height z_i . At this height, the foot contacts the ground but does not penetrate into it. We calculate this using a function of ground height like (3.4).

$$z_{j,i} = \text{terrain}(x_{j,i}) \quad \forall j \in \{\text{front, rear}\} \quad (3.4)$$

Besides sampling foot position (x_i, z_i) , we sample an angle of contact θ_i . Sampling this angle allows to understand how the robot’s legs interact with the terrain. We can use this information to prevent collisions between the robot’s legs and the terrain. The shank extends from the contact point to a knee joint. By defining this angle, we

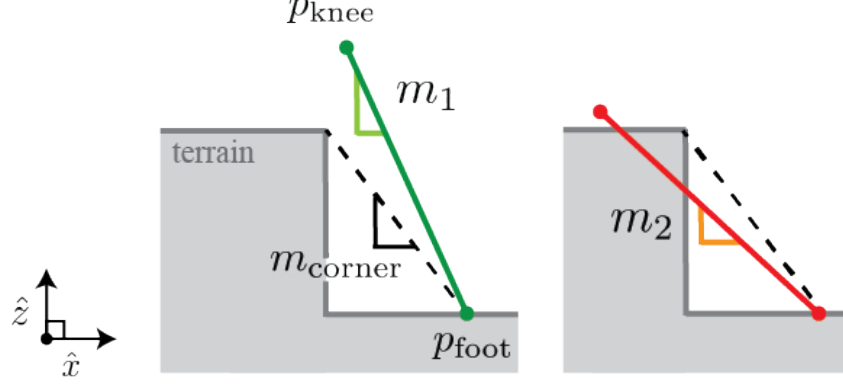


Figure 3-2: To validate the sampled contact angle and prevent collision between the shank and terrain, the slope of the shank link is compared to the discrete terrain corners at position \mathbf{p}_{foot} along the terrain. The left sample is kinematically feasible because $|m_1| > |m_{\text{corner}}|$. The right sample is kinematically infeasible because $|m_2| < |m_{\text{corner}}|$.

calculate the position of that knee joint $\mathbf{p}_{\text{knee},j,i}$ via (3.6).

$$\mathbf{p}_{\text{foot},j,i} = \begin{bmatrix} x_{j,i} \\ z_{j,i} \end{bmatrix} \quad \forall j \in \{\text{front}, \text{rear}\} \quad (3.5)$$

$$\mathbf{p}_{\text{knee},j,i} = \mathbf{p}_{\text{foot},j,i} + l_{\text{shank}} \begin{bmatrix} -\cos(\theta_{j,i}) \\ \sin(\theta_{j,i}) \end{bmatrix} \quad \forall j \in \{\text{front}, \text{rear}\} \quad (3.6)$$

With chosen angle of contact θ_i , we make sure the shank linkage does not collide with the terrain. This can be done in $O(1)$ time by comparing the slope of the shank link against the slope from the contact point to the nearest corner of terrain. If the slope to the terrain corner is steeper, we have a collision across the shank and must sample a new contact position or angle. This conditional is convenient to solve over a discretized terrain where terrain corners are known. Figure 3-2 illustrates this no collision constraint evaluated at different contact angles.

Once we know the contact state will not cause collisions with the terrain, we must check that the robot can configure itself to stand at the sampled contact positions \mathbf{p}_i . From sampled contact positions and angles, we know where the feet and knees of our robot must be. We can solve inverse kinematics to decide where the rest of the robot's body should configure itself to satisfy these positions. However, this problem would be too time consuming to solve in each RRT loop. As an efficient proxy for the inverse kinematics, we can investigate the space between knees to ensure body linkages could connect them. Search spaces for each body angle are displayed in Figure 3-3.

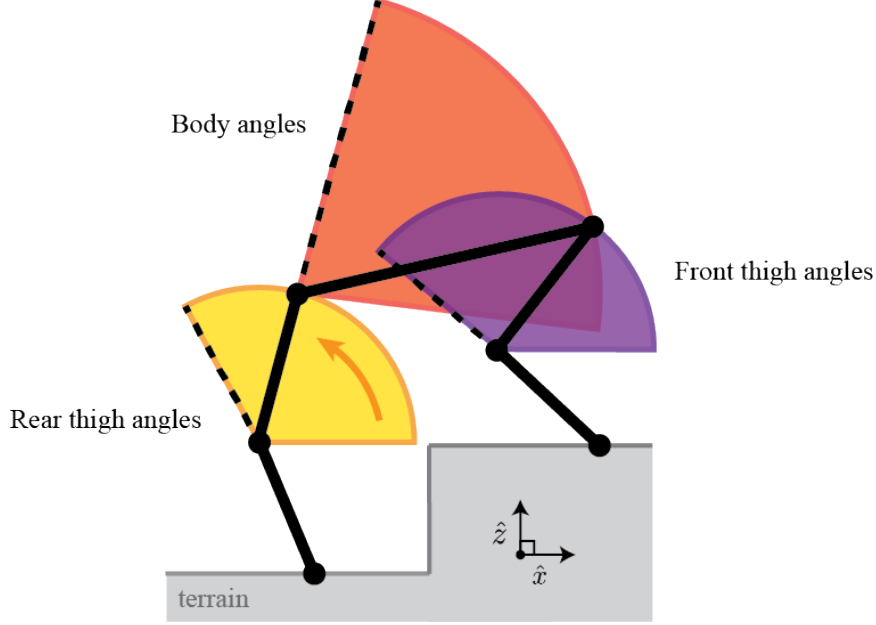


Figure 3-3: Possible body position angles for 2D quadruped given fixed knee positions. The space of possible body angles must expand from the edge of possible rear thigh angles. If the edges of the body and front thigh angle spaces intersect, it is guaranteed that the robot can maintain the knee positions chosen through sampling.

To do run this proxy, a small set of possible rear thigh angles θ_{thigh} are explored around the back knee $\mathbf{p}_{\text{knee,rear},i}$. At each of these possible angles, the distance between the back hip and the front knee $\mathbf{p}_{\text{knee,front},i}$ is measured. If this distance is greater than the difference between the body link length l_{body} and front thigh link length l_{thigh} , then we can ensure that the space of possible body and front thigh angles will overlap. This guarantees our inverse kinematics optimization will be able find a solution for this contact state. We can execute this process in linear time relative to the size of possible rear thigh angles θ_{thigh} using (3.7). By choosing a small, low resolution set of possible rear thigh angles, this can be computed very efficiently. This is a sufficient, but not necessary, condition for inverse kinematic solvability. A similarly efficient feasibility check could be obtained for other robotic systems.

$$\max_{\theta_{\text{thigh},i}} \left\| \mathbf{p}_{\text{knee,rear},i} + l_{\text{thigh}} \begin{bmatrix} \cos(\theta_{\text{thigh},i}) \\ \sin(\theta_{\text{thigh},i}) \end{bmatrix} - \mathbf{p}_{\text{knee,front},i} \right\| \geq l_{\text{body}} - l_{\text{thigh}} \quad (3.7)$$

If any of the above documented kinematic feasibility conditions are not met, the values of contact position \mathbf{p}_i must be resampled until a feasible state is found. Resampling follows the same procedure documented in section 3.2.3 and depends on the value of the last feasible state.

3.2.2 Connecting contact states

If all feasibility checks have been passed, we know the sampled contact state \mathbf{p}_i is kinematically feasible and thus we can add it as a node on our graph G . Once added, we must try to connect this new node to previously sampled contact states. Edges on our graph represent movement between contact states. If a step can be made from one state to the other, we should connect the corresponding graph vertices with an undirected edge. For a step to be taken between two states, only one contact position can change. We can only step one foot at a time, either the front or the rear. This is a consequence of abstracting the problem into 2D. Therefore, states can be connected on our graph when they have only one differing contact point position.

When the contact state \mathbf{p}_i is added to our graph, Euclidean nearest neighbors is ran to find nearby, previously-sampled states \mathbf{p}_{near} . For each of these neighbor states, we investigate if a step could be taken to the current state \mathbf{p}_i . As noted above, states can be stepped between if only one contact positions changes. We evaluate this condition using (3.8).

$$(x_{\text{rear},i} = x_{\text{rear},\text{near}}) \vee (x_{\text{front}} = x_{\text{front},\text{near}}) \quad (3.8)$$

If this condition is satisfied, the states are connected with an unweighted, undirected edge on our graph G . We do not need to do any further checks because we know that each of the individual contact states are independently feasible.

3.2.3 Sampling candidate contact states

After evaluating a feasible contact state \mathbf{p}_i , we would like to sample a new candidate contact state \mathbf{p}_{i+1} that is incrementally closer to our desired goal state. We do this on each iteration of the sampling process. The new state is chosen at random across the terrain along the \hat{x} axis. However, we can employ heuristics to guarantee certain facts about all of the states we sample. By doing this, we improve the efficiency of our algorithm by not wasting time investigating states which are unlikely to be feasible.

We sample a new position \mathbf{p}_{i+1} based on our most recent feasible contact state \mathbf{p}_i . We sample this new position such that our robot could move between these states. This heuristic works to maximize the number of connectable contact states in our graph and ensures we can generate a path from the start to end of the terrain as quickly as possible. Given that we have a 2D system with only two contact points, we require that only one contact position moves between states for them to be con-

nactable. In simplest terms: only one foot can move at a time. Therefore, we sample new contact positions considering constraint (3.9). This way, we know that any sampled contact state could connect to at least one existing contact state. We randomly choose which contact position changes between states, either front or rear, with equal probability.

$$(x_{\text{rear},i} = x_{\text{rear},i+1}) \vee (x_{\text{front}} = x_{\text{front},i+1}) \quad (3.9)$$

Additionally, we should constrain that the robot’s front foot remains ahead of its rear foot in every considered contact state with condition (3.10). This ensures that legs never cross over each other in-between states.

$$x_{\text{rear},i} < x_{\text{front},i} \quad (3.10)$$

Furthermore, the distance between the robot’s feet should be within some desired range in all states. This heuristic works to increase the stability of poses. If feet are too close, the robot’s contact polygon will be small and poses are less stable. Conversely, if the feet are too far apart, the robot may be overstretched. This could lead to complex or unsolvable inverse kinematics. For these reasons, we constrain the distance between foot positions into within a range using a condition like (3.11).

$$d_{\text{min}} \leq x_{\text{front},i} - x_{\text{rear},i} \leq d_{\text{max}} \quad (3.11)$$

We would like our new state to be significantly different than the previous state. It would be wasteful to investigate a state which is approximately the same as the previous state. Thus, we impose a tolerance δ_t between states and require that (3.12) must be satisfied. We choose δ_t to be very small, such that movement between states with difference less than δ_t would require only a tiny, insignificant footstep. In our implementaton, δ_t is defined equal to 0.03 meters.

$$|x_{\text{front},i} - x_{\text{front},i+1}| + |x_{\text{rear},i} - x_{\text{rear},i+1}| > \delta_t \quad (3.12)$$

All of these heuristics considered, we define closed form equations that can be used to sample new candidate positions $x_{\text{front},i+1}$ and $x_{\text{rear},i+1}$. We randomly choose whether to change the front or rear foot position with equal probability. If we choose to move the robot’s front foot, we can calculate new contact positions with (3.13).

This calculation depends on a randomly sampled value r from a uniform distribution.

$$x_{\text{front},i+1} = x_{\text{front},i} + \delta_t + r(d_{\text{max}} - (x_{\text{front},i} - x_{\text{rear},i}) - \delta_t) \quad r \sim U[0, 1] \quad (3.13)$$

$$x_{\text{rear},i+1} = x_{\text{rear},i}$$

If instead we choose to increment the robot’s rear foot, we instead calculate new candidate contact positions with (3.14).

$$\begin{aligned} x_{\text{front},i+1} &= x_{\text{front},i} \\ x_{\text{rear},i+1} &= x_{\text{rear},i} + \delta_t + r(x_{\text{front},i} - x_{\text{rear},i} - 2\delta_t) \quad r \sim U[0, 1] \end{aligned} \quad (3.14)$$

Both (3.13) and (3.14) work to satisfy all aforementioned heuristic conditions. Once new proposed \hat{x} positions have been calculated using these equations, we can find the corresponding \hat{z} position given our map of the terrain. With these coordinates, we can fully describe candidate contact state \mathbf{p}_{i+1} .

Every time we sample a new contact state, a random contact angle θ_{i+1} is chosen for each foot. This enables the robot’s legs to pivot while its foot remains in place. Angles are chosen within a range of values that make sense considering the joint limits of our robot. This works to ensure that we can have a better chance of solving inverse kinematics later. For our implementation, we choose new contact angles θ_{i+1} within the range given by (3.15).

$$\frac{\pi}{8} \leq \theta \leq \frac{\pi}{2} \quad (3.15)$$

3.2.4 Extracting a path

The sampling process is finished once a feasible contact position is sampled near x_{goal} , where x_{goal} is located at the end of the given terrain we’d like to cross. Once this condition is met, a path \mathbf{p} can be traced across the graph of contact states G from this final state to the initial state \mathbf{p}_0 . This search can be conducted through breadth-first search or other simple methods. This gives us a full sequence of the footstep locations that can be used to cross the given terrain.

After the path \mathbf{p} is determined, we can run inverse kinematics on each contact state to extract a sequence of full poses \mathbf{q} for the robot system. This pose sequence is fed into our dynamic optimization as a discrete skeleton trajectory to follow. Calculating a full robot state \mathbf{q}_i for every contact state \mathbf{p}_i gives us a pose sequence of length

N_{sampled} , as represented in (3.16).

$$\mathbf{q} = \left[\mathbf{q}_0 \quad \mathbf{q}_1 \quad \mathbf{q}_2 \quad \dots \quad \mathbf{q}_{N_{\text{sampled}}} \right] \quad (3.16)$$

We run a short kinematic optimization problem, formulated in (3.17), to find full states of the robot \mathbf{q}_i from contact positions \mathbf{p}_i . States are constrained such that the foot and knee positions of the quadruped's configuration must match sampled values. A cost function is designed to find a full state that is as close to statically stable as possible. To do so, we square the difference between the centroidal body position $q_{x,i}$ and the mean foot position. We also attempt to minimize centroidal pitch of the robot ϕ . These terms are weighted such that $\alpha_x > \alpha_\phi$.

$$\begin{aligned} \min_{\mathbf{q}_i} \quad & \alpha_x \left(q_{x,i} - \frac{x_{\text{front},i} - x_{\text{rear},i}}{2} \right)^2 + \alpha_\phi q_{\phi,i}^2 \\ \text{s.t.} \quad & \text{foot_positions}(\mathbf{q}) = [\mathbf{p}_{\text{foot,front},i}, \mathbf{p}_{\text{foot,rear},i}] \\ & \text{knee_positions}(\mathbf{q}) = [\mathbf{p}_{\text{knee,front},i}, \mathbf{p}_{\text{knee,rear},i}] \end{aligned} \quad (3.17)$$

Pseudocode for the full sampling-based footstep planning algorithm is provided in Algorithm 2. This gives a fundamental overview of how the algorithm can be implemented.

Algorithm 2: Generate Kinematically Feasible Path

Input: Starting contact positions \mathbf{p}_0 , Terrain map $\text{terrain}(x)$, Maximum iterations N_{\max}

Output: Footstep sequence \mathbf{p} , Pose sequence \mathbf{q}

```
1 Initialize tree with root:  $G \leftarrow \mathbf{p}_0$ 
2  $\mathbf{p}_{\text{current}} \leftarrow \mathbf{p}_0$ 
3 while  $\mathbf{p}_{\text{current}}$  is not at the end of the terrain do
4    $\mathbf{p}_i \leftarrow \text{SAMPLE\_NEW\_POSITION}(\mathbf{p}_{\text{current}}, \text{terrain})$ 
5   if  $\text{KINEMATICALLY\_FEASIBLE}(\mathbf{p}_i, \text{terrain})$  then
6      $\mathbf{p}_{\text{near}} \leftarrow \text{NEAREST\_NEIGHBOR}(\mathbf{p}_{\text{near}}, G)$ 
7     if  $\text{STEPPABLE}(\mathbf{p}_{\text{near}}, \mathbf{p}_i)$  then
8        $G.\text{add\_vertex}(\mathbf{p}_i)$ 
9        $G.\text{add\_edge}(\mathbf{p}_{\text{near}}, \mathbf{p}_i)$ 
10       $\mathbf{p}_{\text{current}} \leftarrow \mathbf{p}_i$ 
11    end if
12  end if
13  if  $i > N_{\max}$  then
14    break
15  end if
16 end while
17  $\mathbf{p} \leftarrow \text{BFS}(\mathbf{p}_{\text{current}}, \mathbf{p}_0)$ 
18  $\mathbf{q} \leftarrow \text{INVERSE\_KINEMATICS}(\mathbf{p})$ 
```

3.3 Dynamic optimization

After a sequence of contact positions is sampled through our RRT-based algorithm, an optimization problem can be ran to ensure rigorous dynamic feasibility of this proposed trajectory. The Cartesian coordinates of sampled contact positions $\mathbf{p}_{\text{foot},i}$, as defined in (3.18), act as a guiding basis for the optimization problem. These determine the gait and foot positions that the decision variables must optimize over. Using this, contact forces and a continuous centroidal trajectory are found in accordance with sampled contact positions to assess dynamic feasibility. We follow a centroidal approach to our optimization, following what is documented in [9].

$$\mathbf{p}_{\text{foot},i} = \begin{bmatrix} x_{\text{front}} & x_{\text{rear}} \\ z_{\text{front}} & z_{\text{rear}} \end{bmatrix} \quad \forall i \in [0, N_{\text{sampled}} - 1] \quad (3.18)$$

We scale our problem with respect to the number of sampled poses. The total number of timesteps N is calculated to be some constant integer k times the number

of sampled poses from RRT N_{sampled} .

$$N = k(N_{\text{sampled}} - 1) + 1, \quad k \geq 1 \in \mathbb{Z} \quad (3.19)$$

From a high-level view, our decision variables are simply the robot’s center body position and pitch \mathbf{c}_i , the centroidal velocity \mathbf{v}_i , and the contact forces on each foot \mathbf{F}_i at each timestep i .

$$\mathbf{c}_i = \begin{bmatrix} c_x \\ c_z \\ c_\phi \end{bmatrix}, \quad \mathbf{v}_i = \begin{bmatrix} v_x \\ v_z \\ v_\phi \end{bmatrix} \quad \forall i \in [0, N - 1] \quad (3.20)$$

$$\mathbf{F}_i = \begin{bmatrix} F_{x,\text{front}} & F_{x,\text{rear}} \\ F_{z,\text{front}} & F_{z,\text{rear}} \end{bmatrix} \quad \forall i \in [0, N - 1] \quad (3.21)$$

The duration of time between discrete points i and $i + 1$ is optimized as h_i . This timestep is an element within a full decision vector \mathbf{h} outlined by (3.22).

$$\mathbf{h} = [h_0 \quad h_1 \quad h_2 \quad \dots \quad h_{N-2}] \quad (3.22)$$

Solving optimization with these decision variables gives a continuous centroidal trajectory for the robot to follow, and required forces at contact. The optimization is nonlinear and can be solved with popular solvers such as IPOPT or SNOPT [45, 17]. Due to its non-linearity, it is not guaranteed to return an optimal solution with respect to our cost function. At best, it can return a feasible solution that is guaranteed to be at least locally optimal. In practice, we observe such convergence is both frequent and sufficiently fast. If convergence is reached, the trajectory can then be sent to a whole-body controller to control a robot across physical terrain.

3.3.1 Kinematic constraints

First, we impose kinematic constraints on our optimization. By pre-sampling kinematically feasible poses, we have reduced the number of required kinematic constraints. Notably, we have eliminated the need for nonlinear complementarity constraints on foot position. These constraints typically require the use of a highly nonlinear forward kinematics function to ensure that robot feet make proper contact with the ground. We have used our sampling-based algorithm to eliminate the need for this problematic kind of constraint.

However, we still need to impose some simple kinematic constraints on the robot’s

center body position. We would like optimize across the terrain between start and goal positions. The first centroidal position \mathbf{c}_0 is constrained at a chosen starting position $\mathbf{c}_{\text{start}}$ by condition (3.23).

$$\mathbf{c}_0 = \mathbf{c}_{\text{start}} \quad (3.23)$$

The final centroidal position \mathbf{c}_N is constrained to be within a small bounding box of user-chosen size ϵ_{goal} around a chosen end position \mathbf{c}_{goal} by condition (3.24).

$$\mathbf{c}_{\text{goal}} - \begin{bmatrix} \epsilon_{\text{goal}} \\ \epsilon_{\text{goal}} \\ \epsilon_{\text{goal}} \end{bmatrix} \leq \mathbf{c}_N \leq \mathbf{c}_{\text{goal}} + \begin{bmatrix} \epsilon_{\text{goal}} \\ \epsilon_{\text{goal}} \\ \epsilon_{\text{goal}} \end{bmatrix} \quad (3.24)$$

We also constrain that start and end robot positions are statically stable. That is, the robot’s center body position must be between the position of foot contacts at the first and final timesteps. This is formulated into condition (3.25).

$$x_{\text{rear},0} \leq c_{x,0} \leq x_{\text{front},0} \quad (3.25)$$

$$x_{\text{rear},N} \leq c_{x,N} \leq x_{\text{front},N}$$

Over the entire trajectory, we constrain that the height of body position $c_{z,i}$ must remain at least some fixed amount above the terrain with (3.26). This ensures we maintain a practical body height throughout the trajectory. Like all previous kinematic constraints, this is a linear constraint.

$$c_{z,i} \geq \text{terrain}(c_{x,i}) + \epsilon \quad (3.26)$$

As previously noted, there are more optimization timesteps than sampled poses $N > N_{\text{sampled}}$. That is because we must allow legs time to swing between each sampled standing pose. We have k timesteps between each set of poses, as calculated in (3.19). Over this duration, feet can swing across free space and remain out of contact with the ground. We do not directly model leg swing in our optimization, but it could be extracted later using a spline generator between contact locations. The leg that swings is determined based on the difference between sequential sampled poses and our stepping condition from section 3.2.2. If the position of the back foot moves but not the front, then the back leg swings and vice versa. In this way, sampled positions dictate the gait pattern of our robot trajectory.

3.3.2 Dynamic constraints

Most importantly, we impose dynamic constraints on our optimization. These constraints help us ensure resulting trajectories are dynamically feasible. The most simple dynamic constraint is to ensure that the change in our decision variables between timesteps equals their derivative, as outlined in (3.27). This and other integration constraints are nonlinear because timesteps h_i are decision variables.

$$\mathbf{c}_{i+1} = \mathbf{c}_i + h_i \mathbf{v}_i \quad \forall i \in [0, N - 2] \quad (3.27)$$

Timesteps h_i are also constrained to be within a small specified range, following (3.28).

$$h_{\min} \leq h_i \leq h_{\max} \quad \forall i \in [0, N - 2] \quad (3.28)$$

Through kinematic constraints, we have imposed that feet must be in contact with the ground as dictated by the sampled pose sequence. Forces can only be imposed on contact positions when feet are on the ground. We can use constraints like (3.29) and (3.30) to enforce this. If the foot is on the ground, vertical force can be non-zero. Otherwise, there must be no force on the foot. This constraint is still linear with respect to decision variables because foot locations are a pre-determined input parameter to the optimization. In a traditional approach, contact locations would be decision variables. Therefore, this constraint would need to be a nonlinear complementarity constraint, which are notoriously problematic for conventional solvers due to their discontinuous gradients. We have eliminated this need by pre-sampling contact locations.

$$(z_{\text{front},i} - \text{terrain}(x_{\text{front},i}))F_{z,\text{front},i} = 0 \quad \forall i \in [0, N - 1] \quad (3.29)$$

$$(z_{\text{rear},i} - \text{terrain}(x_{\text{rear},i}))F_{z,\text{rear},i} = 0 \quad \forall i \in [0, N - 1] \quad (3.30)$$

Forces on the robot's feet must obey friction. We can ensure this is true by enforcing a friction cone constraint (3.31) that limits the magnitude of force in the \hat{x} direction with respect to vertical force in the \hat{z} direction. Moreover, this constraint ensures that horizontal force is zero wherever vertical force is zero.

$$-\mu \mathbf{F}_{z,j} \leq \mathbf{F}_{x,j} \leq \mu \mathbf{F}_{z,j} \quad \forall j \in \{\text{front}, \text{rear}\} \quad (3.31)$$

Additionally, a physical system cannot apply infinite force on the ground. So, we

must constrain the amount of force that can be generated between the ground and our system to be within a realistic range. This is done by constraint (3.32). In our implementation, we set F_{\max} to be 300N. Also, the ground can only supply vertical force on the robot, so \mathbf{F}_z cannot be negative.

$$0 \leq \mathbf{F}_{z,j} \leq F_{\max} \quad \forall j \in \{\text{front}, \text{rear}\} \quad (3.32)$$

Contact forces dictate the acceleration of the robotic system. A free-body diagram about the robot's centroidal position is drawn to extract equations of motion. This diagram is provided in Figure 3-4.

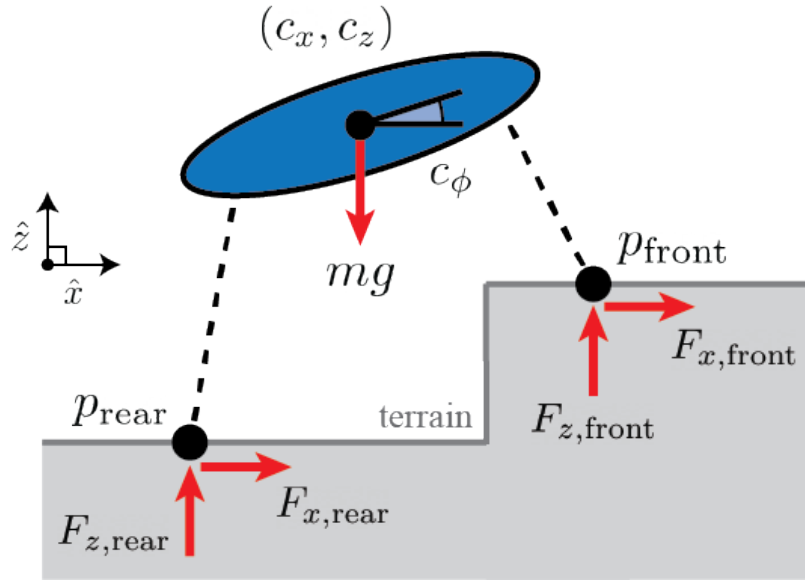


Figure 3-4: Centroidal free-body diagram for the robotic system. Contact forces $F_{x,\text{rear}}, F_{z,\text{rear}}$ are applied at \mathbf{p}_{rear} . And, contact forces $F_{x,\text{front}}, F_{z,\text{front}}$ are applied at $\mathbf{p}_{\text{front}}$. Gravity acts in the $-\hat{z}$ direction from the centroidal position (c_x, c_z) .

From this free body diagram, we sum the forces in each direction to determine the net acceleration of the system. This gives our planar equation of motion as (3.33).

$$\begin{bmatrix} ma_{x,i} \\ ma_{z,i} \end{bmatrix} = \begin{bmatrix} F_{x,\text{rear},i} \\ F_{z,\text{rear},i} \end{bmatrix} + \begin{bmatrix} F_{x,\text{front},i} \\ F_{z,\text{front},i} \end{bmatrix} + \begin{bmatrix} 0 \\ -mg \end{bmatrix} \quad \forall i \in [0, N-1] \quad (3.33)$$

After constraining acceleration through force balance, we constrain the change in velocity in each direction with discrete integration in (3.34).

$$\begin{bmatrix} v_{x,i+1} \\ v_{z,i+1} \end{bmatrix} = \begin{bmatrix} v_{x,i} \\ v_{z,i} \end{bmatrix} + h_i \begin{bmatrix} a_{x,i} \\ a_{z,i} \end{bmatrix} \quad \forall i \in [0, N-2] \quad (3.34)$$

We must consider rotational dynamics too. We can calculate the moments about the center of the robot’s body by considering contact locations and forces. This is formalized by (3.35). This constraint is nonlinear because it requires multiplying decision variables \mathbf{c}_i and \mathbf{F}_i .

$$\begin{aligned} I\dot{\omega}_i = & (x_{\text{front},i} - c_{x,i})F_{z,\text{front},i} - (z_{\text{front},i} - c_{z,i})F_{x,\text{front},i} + \\ & + (x_{\text{rear},i} - c_{x,i})F_{z,\text{rear},i} - (z_{\text{rear},i} - c_{z,i})F_{x,\text{rear},i} \quad \forall i \in [0, N - 1] \end{aligned} \quad (3.35)$$

Contact forces can induce changes in robot body pitch c_ϕ over time. We constrain the integration of rotational motion with (3.36).

$$v_{\phi,i+1} = v_{\phi,i} + h_i\dot{\omega}_i \quad \forall i \in [0, N - 2] \quad (3.36)$$

3.3.3 Cost function

We would like our resulting trajectory to be dynamically feasible and high quality. Imposed constraints guarantee that our resulting trajectory obeys the modeled dynamics. We designate a cost function (3.37) that works to ensure the trajectory is high quality. Our trajectory should match sampled poses as closely as possible. So, we minimize the difference between sampled centroidal positions $\mathbf{c}_{\text{sampled},i}$ and \mathbf{c}_i . Sampled centroidal positions are found through the inverse kinematics optimization (3.17) ran for each set of footstep positions. Additionally, we minimize the difference between \mathbf{v}_i and some reference velocity \mathbf{v}_{ref} . This encourages the velocity of the robot to be nearly uniform. Lastly, we minimize the squared norm of contact forces to encourage energy efficiency. Each of these elements is summed and scaled by preference in the cost function. Exact values of coefficient weights for our implementation are detailed in section 4.1.

$$\begin{aligned} \min \sum_{i=1}^N & (\mathbf{c}_i - \mathbf{c}_{\text{sampled},i})^T \mathbf{Q}_c (\mathbf{c}_i - \mathbf{c}_{\text{sampled},i}) + \\ & + (\mathbf{v}_i - \mathbf{v}_{\text{ref}})^T \mathbf{Q}_v (\mathbf{v}_i - \mathbf{v}_{\text{ref}}) + \mathbf{F}_i^T \mathbf{Q}_F \mathbf{F}_i \end{aligned} \quad (3.37)$$

Chapter 4

Results

We have developed a decoupled planning methodology for a 2D quadruped robotic system. We would like to evaluate the effectiveness of this novel approach. To do so, we implemented the approach in a simulated environment. Once implemented, we tested it in comparison to traditional trajectory optimization approaches. In this chapter, we will provide a detailed explanation of how the methodology was implemented and share results to showcase its relative performance.

4.1 Setup

The described approach to decoupled planning is implemented in Matlab code using the Casadi optimization package [1]. Given a discretized map of a terrain, a sequence of contact positions can be found using our sampling-based algorithm. Then, our dynamic optimization considers these contact positions and determines a centroidal trajectory, using the Casadi package to solve the nonlinear program.

Terrains are represented as 2D discrete maps. All terrains have a constant length L in the \hat{x} direction. Along this length, they are broken into 5 distinct sections. The initial, middle, and final sections are designated to have a constant height of 0 in the \hat{z} direction. The 2nd and 4th sections have a randomized height and width. These sections are meant to represent obstacles. These obstacles can either have a positive height, acting as a step in the terrain. Or, they can have a negative height and act as a valley. Terrain generation parameters are illustrated in Figure 4-1. The range of possible heights is determined based on the size of the robot itself. Increasing this range directly increases the difficulty of crossing the terrain. Flat ground is, of course, the least complex and easiest to plan across.

We plan across terrains for a 2D quadruped system that emulates the MIT Mini-

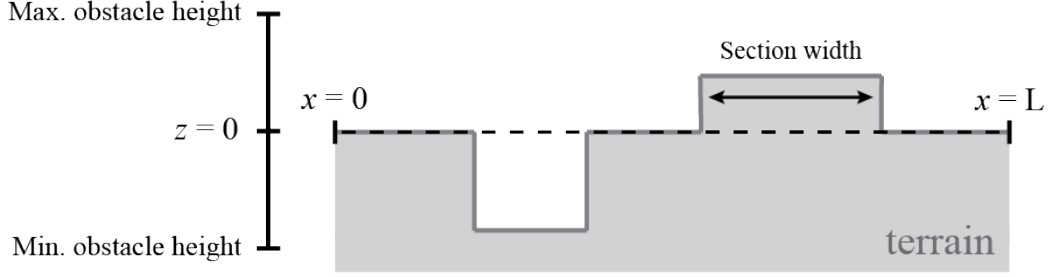


Figure 4-1: Example of a generated terrain. The terrain has a fixed length L in the \hat{x} direction. The initial, middle, and final sections have a constant height of 0 in the \hat{z} direction. The 2nd and 4th sections are designated as obstacles. These sections have randomized heights and widths within a specified possible range.

Cheetah [22]. The simplified robotic system is depicted in Figure 2-1. It acts as a 5-link system with angle limits between each joint. The lengths of body linkages and joint limits specific to our implementation are tabulated in Table 4.1.

Table 4.1: Robot System Parameters

Variable	Value
Body length	0.380 m
Thigh length	0.209 m
Shank length	0.195 m
q_4 limits	$(0, \pi)$
q_5 limits	$(0, \pi)$
q_6 limits	$(0, \pi)$
q_7 limits	$(0, \pi)$

In our dynamic optimization cost function (3.37), each term is assigned a weighting through coefficient matrices. We are most concerned with the centroidal position of the robot relative to sampled poses, thus the corresponding coefficient \mathbf{Q}_e should have the largest weighting in the cost function. Furthermore, we care more about tracking the \hat{z} position of the robot than the \hat{x} centroidal component. This is because the \hat{x} position component depends on the traversal speed of the robot, which is not guaranteed to remain consistent across the trajectory. Considering this fact, we should make the \hat{z} component of \mathbf{Q}_e largest. All implemented weights are defined in Table 4.2.

Table 4.2: Cost Function Weights

Variable	Value
\mathbf{Q}_c	diag(2, 6, 4)
\mathbf{Q}_v	diag(1, 2, 2)
\mathbf{Q}_F	$0.1\mathbf{I}$

4.1.1 Example trajectory generation

Trajectories are planned for a 2D quadruped system over generated terrains. Given a map of the terrain, contact positions are sampled from the beginning to the end of the terrain. A starting position is designated such that each shank makes contact at an angle of $\theta = \pi/4$ with the rear foot at the beginning of the terrain. New contact positions \mathbf{p}_i are sampled until the front foot reaches the end of the terrain. If the search algorithm fails to reach the end condition within the maximum number of iterations, it will be terminated. However, assuming that the end condition is reached, a sequence of contact positions can be sampled from start to finish, and inverse kinematics can be used to find a full state of the robot \mathbf{q}_i for each of these contact positions. This process follows that which is documented in the previous chapter. Figure 4-2 shows an example sequence of poses that were found over a generated terrain.

Once sampled poses have been determined, we have a discrete sequence of centroidal positions. These positions are input into the cost function of the dynamic optimization, allowing the optimized, continuous centroidal trajectory to be fit along these discrete, sampled points. The cost function for the dynamic optimization is detailed in (3.37). Optimized and sampled centroidal trajectories are plotted in Figure 4-3. The sampled position in this graph are the same from Figure 4-2. As expected, the optimized trajectory approximately follows the sampled positions. Some deviation is observed but this is likely necessary to satisfy the dynamic constraints of the system.

In our dynamic optimization, we seek to minimize the robot’s body angle, or pitch ϕ . By doing so, we can enhance the stability of our trajectory and increase the likelihood of successfully executing it on a physical system. Body angle minimization is encoded in our optimization cost function (3.37). The pitch ϕ across our example trajectory is plotted in Figure 4-4. As seen in this graph, the robot’s pitch is very small throughout the optimized trajectory.

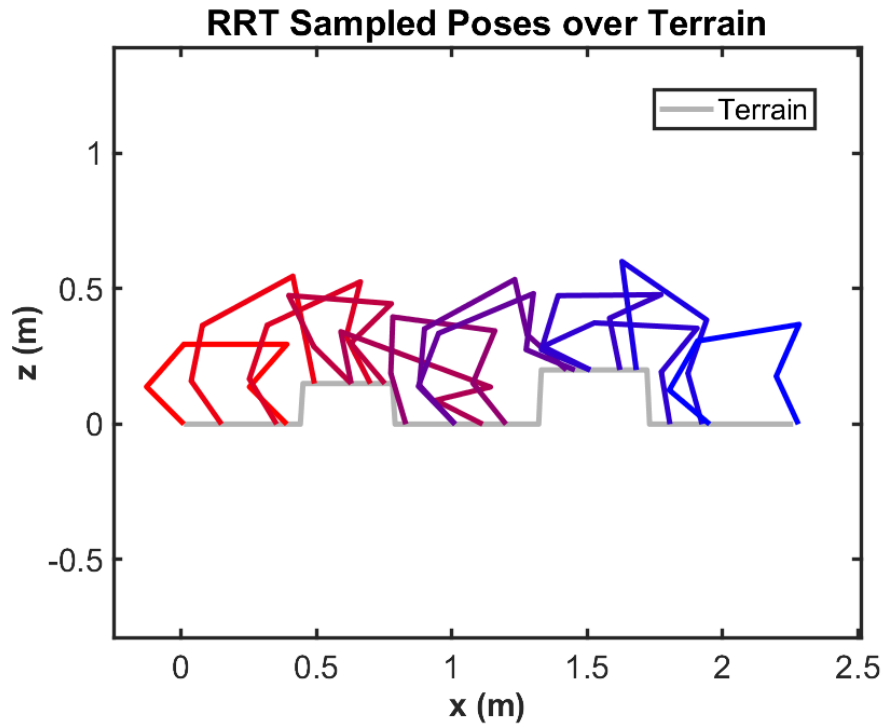


Figure 4-2: A sequence of robotic poses over a generated terrain with obstacle heights of 0.2m and 0.1m. Foot and knee positions are found through our RRT-based sampling algorithm. The corresponding full robotic states are then obtained through inverse kinematics. To reduce clutter in the image, only a few sampled poses from the discrete trajectory are displayed. The color of plotted robot positions change from red to blue as they move from the initial to the final state.

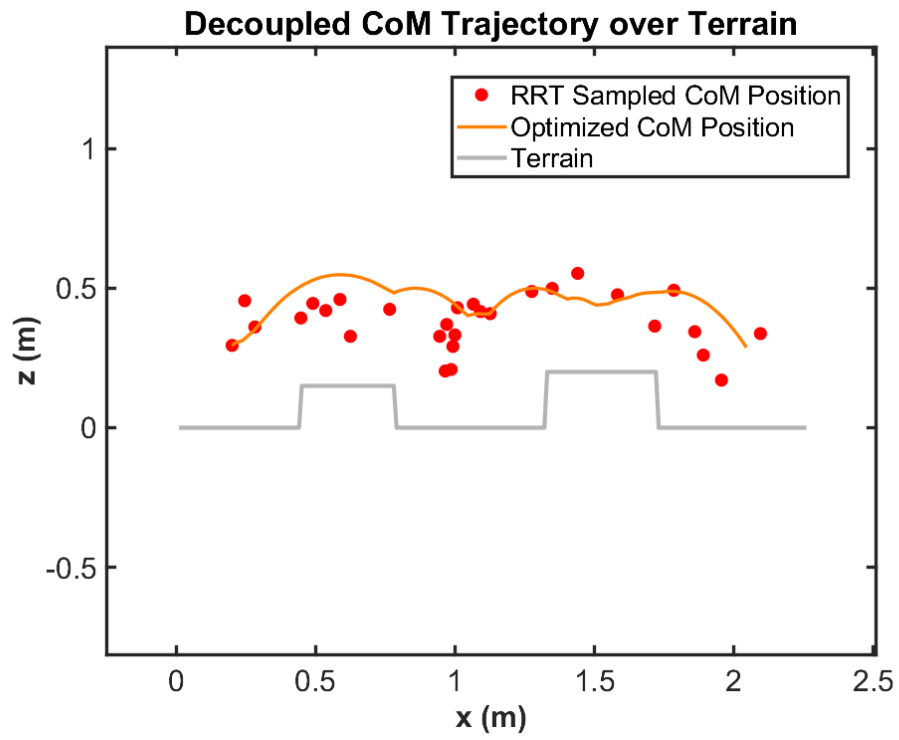


Figure 4-3: The planned centroidal trajectory of the robot's body is plotted over a generated terrain at both stages of the planning process. Discrete centroidal body positions are extracted from sampled contact positions through inverse kinematics. These positions are then used in cost function of dynamic optimization. The continuous, optimized trajectory is displayed in relation to these sampled points.

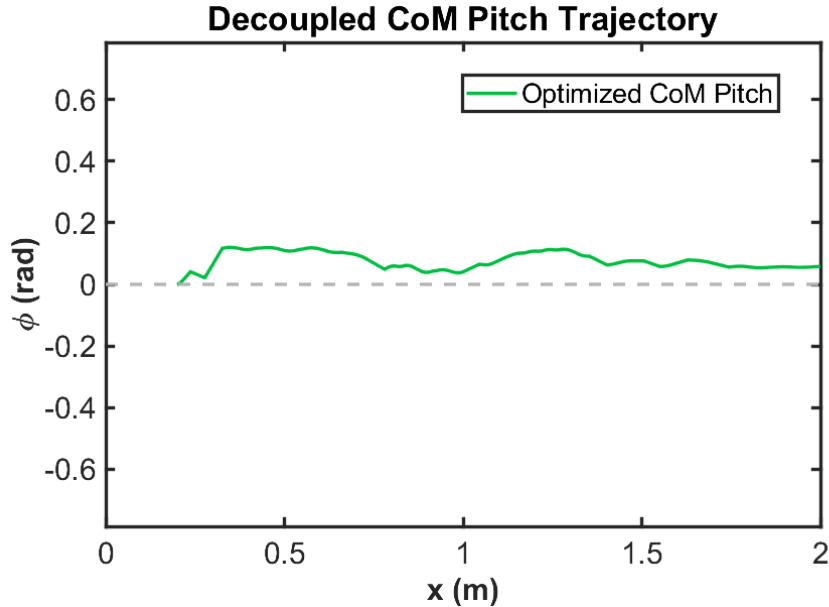


Figure 4-4: The angle about the center of the robot’s main body linkage is shown over the duration of the same trajectory shown in previous graphs. This pitch angle ϕ is shown to be nearly zero across the trajectory.

4.2 Benchmark against coupled approach

We hypothesize that our decoupled approach can be used to generate dynamically feasible trajectories faster than a traditional approach where kinodynamic constraints are coupled into one optimization problem. That is because a decoupled optimization uses RRT to quickly find contact positions, instead of including them as decision variables in a large optimization problem. To test our hypothesis and compare the two approaches, a coupled approach was implemented in Matlab using Casadi. This benchmarking code was developed to follow previously documented implementations [9].

Traditional trajectory optimization formulations have a difficult time choosing footstep positions over a discrete terrain. So, for our benchmark coupled approach, we modeled terrain as a continuous function of ground height such that $z = \text{terrain}(x)$. To approximate a discrete step, this function took on the form $z = a \tanh(bx + c)$. This allowed us to directly encode a map of the ground height into constraints without the need for tedious conditional statements. We needed this continuous terrain map to generate solutions because of the coupled approach’s nonlinear complementarity constraints.

A centroidal approach is taken to the coupled optimization problem. Most of the

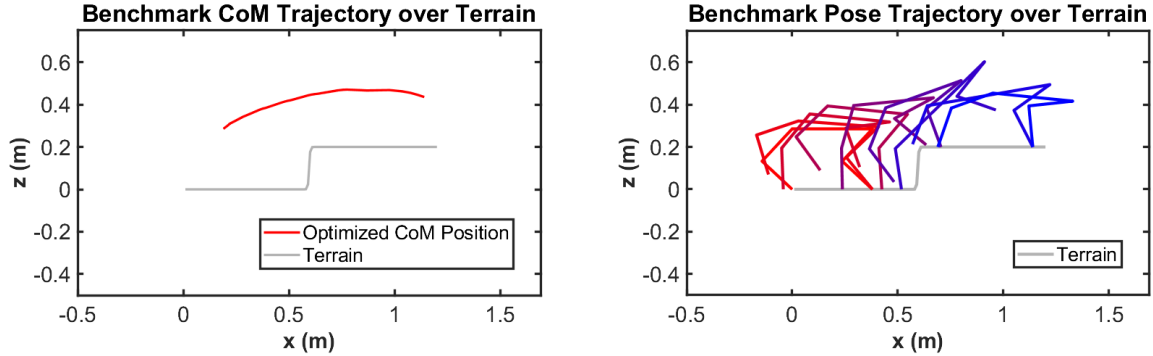


Figure 4-5: The trajectory obtained from the benchmark coupled optimization is plotted over a step terrain that increases in height from 0 to 0.2 meters. The terrain is fully described by (4.1). The continuous centroidal trajectory is displayed along with a selected subset of full robot poses sampled along the trajectory. The robot positions transition from red to blue as they move from the initial to the final state.

constraints directly follow from our decoupled dynamic optimization. However, foot placements along the trajectory must now be chosen through optimization, which necessitates the inclusion of contact-based complementarity constraints. These problematic constraints are avoided in our decoupled implementation by pre-sampling contact locations. In this case, complementarity constraints are nonlinear because footstep locations are decision variables. To simplify the coupled optimization, a gait schedule is provided to ensure the robot steps one foot at a time, though the locations of these steps still need to be determined. Without this gait schedule, the optimization did not reliably converge to feasible solutions. Figure 4-5 displays an optimized trajectory over a 0.2m high step, with the terrain in this example described by (4.1). Here, we see that the coupled approach is capable of finding a feasible full state trajectory over the continuous terrain. Some select poses are plotted to show the movement of the robot from initial to final state.

$$z = \text{terrain}(x) = 0.1 \tanh(100(x - 0.6)) + 0.1 \quad (4.1)$$

With a coupled implementation working, we can compare the speed it takes to find a feasible trajectory against that of our decoupled implementation. We compare average solve times over a completely flat terrain and a continuous step terrain that is identical to that shown in Figure 4-5. These terrains are chosen because they are convenient for the benchmark coupled approach to generate solutions across.

As expected, we observe that the decoupled approach can generate feasible trajectories much faster than the coupled approach. This decrease in solve time is due

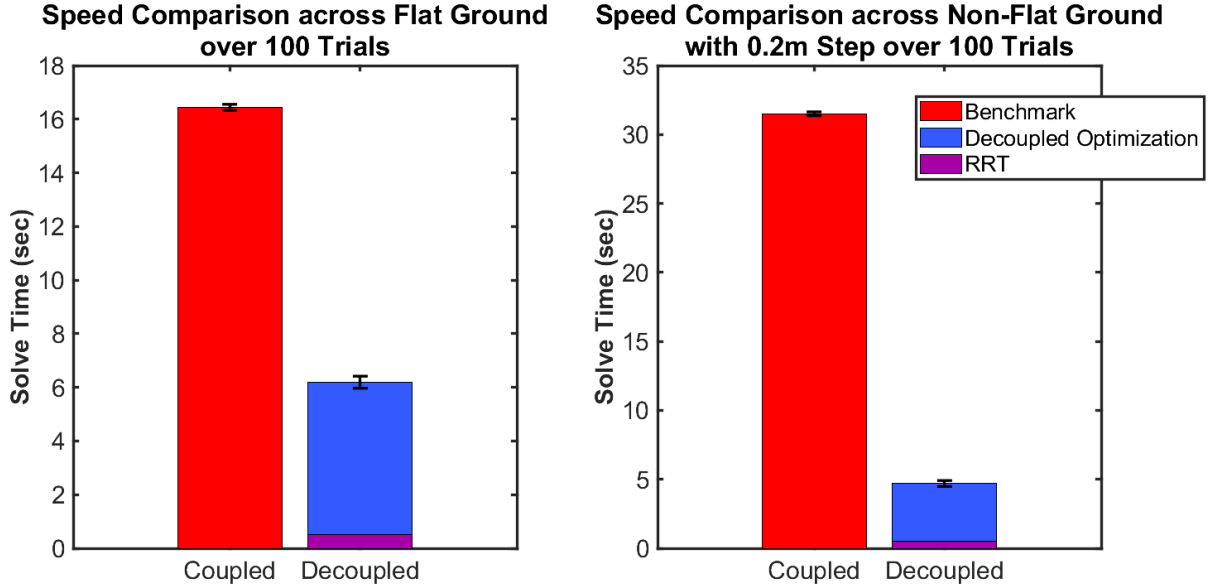


Figure 4-6: Average solve times are presented with standard error bounds for the coupled and decoupled approaches over 100 trials on both flat and non-flat terrains. The length of each terrain is 1.2m, and the non-flat terrain features approximates a discrete step up as defined by (3.4).

to the reduction of our dynamic optimization problem. By using the decoupled approach, we do not have footstep locations as decision variables and thus eliminate the need for nonlinear complementarity constraints. Removing these decision variables and constraints allows the solver to find a solution much faster, as demonstrated by Figure 4-6.

The difference in speed between approaches changes with complexity of terrain. In the case of flat ground, the decoupled approach can compute a trajectory more than twice as fast. But, in the non-flat case, the decoupled approach is more than 6 times as fast. This discrepancy can be attributed to the fact that hybrid-dynamic optimization problems become more difficult over a complex discrete terrain. With discrete obstacles, the gradient of ground height can assume very large values or be undefined at points. This makes evaluating complementarity constraints on contact positions more challenging. Thus, coupled approaches need more optimization iterations to consider possibilities within the decision space of contact positions. In contrast, sampling-based approaches are not affected by discrete obstacles, due to their stochasticity. Regardless of whether the terrain is flat or more complex, positions are sampled across its length. Once we have these positions, the optimization is a lot simpler. We have leveraged these facts to find a dynamically feasible trajectory

Table 4.3: Terrain Difficulty

Difficulty Level	Relative Height	Obstacle Height Limits
0	0	(0, 0) m
1	$\pm(1/5)$ (Body length)	(-0.076, 0.076) m
2	$\pm(2/5)$ (Body length)	(-0.152, 0.152) m
3	$\pm(3/5)$ (Body length)	(-0.228, 0.228) m
4	$\pm(4/5)$ (Body length)	(-0.304, 0.304) m
5	\pm (Body length)	(-0.380, 0.380) m

over complex terrain in less than 15% of the traditionally required time.

It is worth noting that the size of coupled and decoupled optimization problems scale differently. The number of decision variables in the implemented coupled approach is directly proportional to a user chosen variable N_{coupled} . By increasing this value, the number of poses across the terrain is increased. In the decoupled approach, the number of decision variables is directly proportional to the number of poses sampled in RRT. The optimization must account for how the robot’s legs can swing between each static sampled pose. So, our optimization size is proportional to the number of sampled poses N_{sampled} , as documented in (3.19). The user cannot control the size of the decoupled optimization problem because poses are sampled randomly through RRT. We can expect that with increasing terrain length, more poses will be sampled and solve time may increase. However, we can also expect that a longer terrain would analogously require a higher N_{coupled} in the coupled approach.

4.3 Performance analysis

We want to ensure that the decoupled approach is robust and scalable over a wide variety of terrains. To test this, we run our implementation over sets of generated terrains with increasing difficulty. Difficulty is defined to increase with the maximum possible size of obstacles. We assume that terrains with larger obstacles are harder to traverse and plan across. Levels of terrain difficulty are defined relative to the body length of our 2D robotic system, as detailed in Table 4.3. These levels of difficulty do not exceed the body length of the robot because sampling kinematically feasible poses would become more challenging if obstacles were taller than the robot. In a real-world scenario, the robot may traverse taller obstacles by jumping onto them rather than stepping onto them.

Having defined tiers of terrain difficulty, we proceeded to evaluate our implementation in each of these tiers. Across 100 trials in each difficulty tier, we found that the performance remained consistent. Solve times and success rates are plotted in Figure 4-7. Notably, the implementation required approximately 8 seconds to identify a feasible solution, regardless of terrain difficulty. This proves that our novel methodology is robust to many kinds of terrains. Considering previous results, this implies we can generate trajectories much faster than traditional approaches in any of these cases.

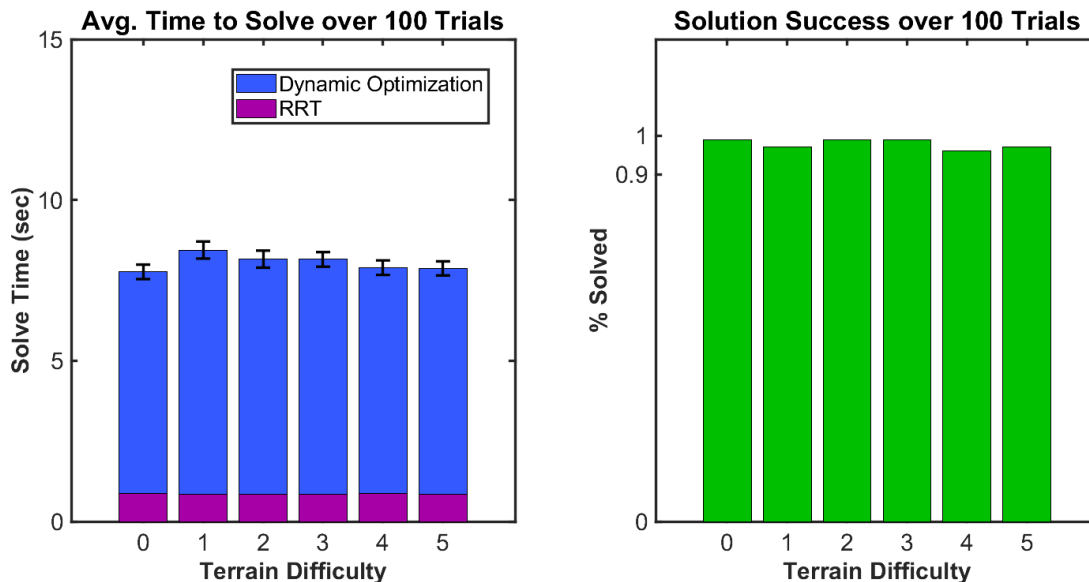


Figure 4-7: Our decoupled implementation was ran over a set of terrains with varying degrees of complexity. The average time needed to find a path across each of these generated terrain sets is plotted with standard error bounds. Bars are divided based on the time consumed by each subprocess involved in the planning approach, namely RRT-based contact sampling and dynamic optimization. The success rate of our overall implementation across these terrains is plotted as well. All generated terrains had a constant total length of 2.25m.

Additionally, we see that the decoupled implementation achieved a success rate of nearly 100% across all levels of difficulty. This success rate slightly decreased as we approached the most challenging sets of terrains, where kinematically feasible poses become more difficult to sample. In a real world implementation, the decoupled architecture should be ran on a loop. If a feasible path cannot be identified, the process should be rerun. Given that RRT is a randomized algorithm, the likelihood of identifying a feasible path in subsequent executions is equally high. This proposed architecture should alleviate any concerns regarding the inability to identify a feasible solution.

Chapter 5

Conclusion

In this thesis, a novel approach to robotic path planning has been introduced. This approach leverages advantages from established sampling-based and optimization-based planning methods to effectively decouple kinodynamic constraints and provide dynamically feasible trajectories at increased efficiency. An RRT-based approach is used to identify a sequence of kinematically feasible contact positions, or simply footsteps, across the terrain. These contact positions are then fed into a dynamic optimization problem which determines a dynamically feasible centroidal trajectory and necessary contact forces. A solution to this optimization can be obtained in less time than required by traditional, coupled optimization methods that must include contact positions in their decision space. Both traditional and decoupled approaches are compared across terrains in a 2D environment. Tests show that the novel decoupled approach can generate a kinodynamically feasible trajectory in less than 15% of the time needed for a traditional, coupled approach. We can attribute this increase in speed to a reduction in our optimization problem. By sampling contact positions ahead of time, we can remove them as decision variables and no longer need to impose problematic complementarity constraints. This directly impacts solving efficiency.

Our decoupled approach has the additional advantage of generating emergent gait patterns. We are restricted to a bounding gait in the sense that the front legs are treated as a pair that must move together and the rear legs as a separate pair. This is consequence of abstracting the problem to 2D. However, the location, order, and timing of how these pairs move is emergent. All footstep locations are found through sampling, and thus the order or size of steps are purely random. Traditional, coupled approaches often require a prescribed gait schedule to speed the optimization process. This can limit the ability of the planner.

To verify the feasibility of the decoupled approach over a diverse range of terrains,

we conducted additional tests on sets of randomly generated terrains with varying levels of complexity. Our results indicate that the novel decoupled approach performs consistently, regardless of what terrain provided. Specifically, a viable trajectory was successfully determined for almost every generated terrain, and the solving time remained comparable across all tested terrains. This outcome demonstrates the robustness of the decoupled method and quality of trajectories generated. This behavior can be attributed to the effectiveness of our heuristics and the stochastic nature of our footstep sampling method. By randomly sampling footsteps, we have the ability to adjust our gait to the terrain as required and our heuristics allow us to do so intelligently.

Beside its successful performance, the decoupled approach is remarkably easy to implement, as demonstrated through the simple problem formulations utilized throughout this paper. Our approach employs a basic version of the widely-used RRT algorithm to sample contact positions. Moreover, our dynamic optimization problem is reduced in size compared to conventional trajectory optimization methods, making it even easier to implement than these classical formulations.

5.1 Future work

Although the novel decoupled method for path planning has exhibited strong performance in a simulated 2D environment, its robustness in a real-world setting remains untested. In future work, we plan to utilize our decoupled implementation to generate centroidal trajectories across actual terrains. These trajectories can then be fed into a whole-body controller, that can transform them into an executable, continuous trajectory. This experiment would allow us to validate the practicality of the decoupled technique and confirm that its performance effectively translates outside of simulated environments.

Furthermore, the described formulation assumes a 2D simplification of the quadruped system. This inhibits the range of trajectories that can be generated. Most notably, it requires that we move the front feet together and the rear feet together in a bounding-style gait. To address this limitation, we could modify the problem formulation to include the complete 3D version of the system. That way, we will have full expressiveness of the system in our planning approach. Additionally, this modification would allow us to model and account for real-world terrains that may not be flat in the \hat{y} direction.

If these developments go well, the decoupled methodology could even be applied

to generate plans for other hybrid-dynamic systems, such as a two-legged humanoid robot. Exact constraints and system dynamics would have to change for this to work, but the fundamental approach would remain largely consistent.

Bibliography

- [1] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, March 2019.
- [2] John T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, March 1998.
- [3] Ricard Bordalba, Lluís Ros, and Josep M. Porta. Randomized Kinodynamic Planning for Constrained Systems. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7079–7086, Brisbane, QLD, May 2018. IEEE.
- [4] Stephen P. Boyd and Lieven Vandenbergh. *Convex optimization*. Cambridge University Press, Cambridge, UK ; New York, 2004.
- [5] M.S. Branicky, M.M. Curtiss, J.A. Levine, and S.B. Morgan. RRTs for nonlinear, discrete, and hybrid planning and control. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 1, pages 657–663, Maui, Hawaii, USA, 2003. IEEE.
- [6] Xu Chang, Hongxu Ma, and Honglei An. Quadruped Robot Control through Model Predictive Control with PD Compensator. *International Journal of Control, Automation and Systems*, 19(11):3776–3784, November 2021.
- [7] Matthew Chignoli, Savva Morozov, and Sangbae Kim. Rapid and Reliable Quadruped Motion Planning with Omnidirectional Jumping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6621–6627, Philadelphia, PA, USA, May 2022. IEEE.
- [8] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.
- [9] Hongkai Dai, Andres Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302, Madrid, November 2014. IEEE.

- [10] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 279–286, Madrid, Spain, November 2014. IEEE.
- [11] Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bleedt, and Sangbae Kim. Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Madrid, October 2018. IEEE.
- [12] Yanran Ding, Mengchao Zhang, Chuanzheng Li, Hae-Won Park, and Kris Hauser. Hybrid Sampling/Optimization-based Planning for Agile Jumping Robots on Challenging Terrains. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2839–2845, Xi’an, China, May 2021. IEEE.
- [13] Mohamed Elbanhawi and Milan Simic. Sampling-Based Robot Motion Planning: A Review. *IEEE Access*, 2:56–77, 2014.
- [14] Maurice Fallon, Scott Kuindersma, Sisir Karumanchi, Matthew Antone, Toby Schneider, Hongkai Dai, Claudia Pérez D’Arpino, Robin Deits, Matt DiCicco, Dehann Fourie, Twan Koolen, Pat Marion, Michael Posa, Andrés Valenzuela, Kuan-Ting Yu, Julie Shah, Karl Iagnemma, Russ Tedrake, and Seth Teller. An Architecture for Online Affordance-based Perception and Whole-body Planning: An Architecture for Online Affordance-based Perception. *Journal of Field Robotics*, 32(2):229–254, March 2015.
- [15] Farbod Farshidian, Michael Neunert, Alexander W. Winkler, Gonzalo Rey, and Jonas Buchli. An Efficient Optimal Planning and Control Framework For Quadrupedal Locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 93–100, May 2017. arXiv:1609.09861 [cs].
- [16] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer US, Boston, MA, 2008.
- [17] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1):99–131, January 2005.
- [18] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive Locomotion through Nonlinear Model Predictive Control. 2022. Publisher: arXiv Version Number: 1.
- [19] Dong Jin Hyun, Sangok Seok, Jongwoo Lee, and Sangbae Kim. High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the MIT Cheetah. *The International Journal of Robotics Research*, 33(11):1417–1445, September 2014.

- [20] L. Jaillet, J. Hoffman, J. van den Berg, P. Abbeel, J. M. Porta, and K. Goldberg. EG-RRT: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2646–2652, San Francisco, CA, September 2011. IEEE.
- [21] Sertac Karaman and Emilio Frazzoli. Sampling-based Algorithms for Optimal Motion Planning, May 2011. arXiv:1105.1186 [cs].
- [22] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301, Montreal, QC, Canada, May 2019. IEEE.
- [23] Matthew Kelly. An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. *SIAM Review*, 59(4):849–904, January 2017.
- [24] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bleedt, and Sangbae Kim. Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control. 2019. Publisher: arXiv Version Number: 1.
- [25] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001, San Francisco, CA, USA, 2000. IEEE.
- [26] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, March 2016.
- [27] Scott Kuindersma, Frank Permenter, and Russ Tedrake. An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2589–2594, May 2014. arXiv:1311.1839 [cs].
- [28] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. 1998.
- [29] Steven M. LaValle and James J. Kuffner. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400, May 2001.
- [30] He Li and Patrick M. Wensing. Hybrid Systems Differential Dynamic Programming for Whole-Body Motion Planning of Legged Robots. *IEEE Robotics and Automation Letters*, 5(4):5448–5455, October 2020.

- [31] Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. Sampling-based contact-rich motion control. *ACM Transactions on Graphics*, 29(4):1–10, July 2010.
- [32] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):1–8, August 2012.
- [33] Michael Neunert, Markus Stauble, Markus Gifftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, July 2018.
- [34] Jorge Nocedal and Stephen J. Wright, editors. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, 1999.
- [35] Brahayam Ponton, Majid Khadiv, Avadesh Meduri, and Ludovic Righetti. Efficient Multicontact Pattern Generation With Sequential Convex Approximations of the Centroidal Dynamics. *IEEE Transactions on Robotics*, 37(5):1661–1679, October 2021.
- [36] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, January 2014.
- [37] Michael Posa and Russ Tedrake. Direct Trajectory Optimization of Rigid Body Dynamical Systems through Contact. In Emilio Frazzoli, Tomas Lozano-Perez, Nicholas Roy, and Daniela Rus, editors, *Algorithmic Foundations of Robotics X*, volume 86, pages 527–542. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. Series Title: Springer Tracts in Advanced Robotics.
- [38] A. Shkolnik, M. Walter, and R. Tedrake. Reachability-guided sampling for planning under differential constraints. In *2009 IEEE International Conference on Robotics and Automation*, pages 2859–2865, Kobe, May 2009. IEEE.
- [39] Alexander Shkolnik, Michael Levashov, Ian R. Manchester, and Russ Tedrake. Bounding on rough terrain with the LittleDog robot. *The International Journal of Robotics Research*, 30(2):192–215, February 2011.
- [40] Russ Tedrake. *Underactuated Robotics*. 2023.
- [41] Steve Tonneau, Andrea Del Prete, Julien Pettre, Chonhyon Park, Dinesh Manocha, and Nicolas Mansard. An Efficient Acyclic Contact Planner for Multi-ped Robots. *IEEE Transactions on Robotics*, 34(3):586–601, June 2018.

- [42] Alexander W. Winkler, C. Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, July 2018.
- [43] David Wooden, Matthew Malchano, Kevin Blankespoor, Andrew Howardy, Alfred A Rizzi, and Marc Raibert. Autonomous navigation for BigDog. In *2010 IEEE International Conference on Robotics and Automation*, pages 4736–4741, Anchorage, AK, May 2010. IEEE.
- [44] Albert Wu, Sadra Sadraddini, and Russ Tedrake. R3T: Rapidly-exploring Random Reachable Set Tree for Optimal Kinodynamic Planning of Nonlinear Hybrid Systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4245–4251, Paris, France, May 2020. IEEE.
- [45] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, March 2006.