Online Receding Horizon Planning of Multi-Contact Locomotion

Jiayi Wang

Doctor of Philosophy
School of Informatics
University of Edinburgh

2023

Jiayi Wang:
Online Receding Horizon Planning of Multi-Contact Locomotion
Doctor of Philosophy, 2023

SUPERVISORS:
Prof. Sethu Vijayakumar, Ph.D., FRSE
Dr. Steve Tonneau, Ph.D.

EXAMINERS:
Dr. Ludovic Righetti, Ph.D.
Dr. rer. nat. Michael Herrmann, Ph.D.

LAY SUMMARY

Legged robots such as humanoids and quadrupeds can overcome terrain irregularities by using multiple contacts between their limbs and the environment. Such capability makes legged robots particularly useful for tasks taking place in unstructured environments, e.g., exploring disaster zones or inspecting construction sites.

Nevertheless, when deploying legged robots in the real world, legged robots can encounter unexpected disturbances, such as dynamic changes of the environment or state deviations induced by external force perturbations. These disturbances can cause the pre-planned motion to become invalid, and continuing to execute the current motion plan will cause the robot to fall. To achieve reliable operation under these disturbances, it is important that legged robots can online (re)-plan their motions.

To achieve this goal, Receding Horizon Planning (RHP) can be a promising solution. RHP refers to the planning strategy that aims to constantly update the motion plan for immediate execution based on the state of the robot and the environment. Typically, to ensure the motion planned for execution can benefit the future operation, RHP frameworks often need to consider an extended planning horizon, which incorporates: 1) an execution horizon that plans the motion to be executed; 2) a prediction horizon that foresees the future. Nevertheless, given the combinatorial nature of the contact planning problem (determining the sequence in which the limbs break and make contact with the ground), and the high-dimensionality and non-convexity of the robot dynamics, computing multi-contact motions over a long planning horizon is often a time-consuming process. Such computational challenge often prevents us from planning multi-contact motions in an online RHP fashion.

To facilitate online multi-contact RHP, in this thesis, we focus on developing novel methods to accelerate the computation speed by finding valid simplifications to the original planning problem. To this end, we propose to introduce model simplifications along the planning horizon. Furthermore, we also explore the usage of machine learning techniques to encode domain knowledge that can be used to bootstrap the computation. We evaluate the computation performance of our methods with rigorous simulation studies. Additionally, we also validate our approach with real-world robot experiments on the humanoid robot Talos and the quadruped robot ANYmal. Thanks to the improved computation efficiency, we successfully demonstrate online multi-contact RHP in real-world experiments that consider dynamic changes in the environment.

ABSTRACT

Legged robots can traverse uneven terrain by using multiple contacts between their limbs and the environment. Nevertheless, to enable reliable operation in the real world, legged robots necessarily require the capability to online re-plan their motions in response to changing conditions, such as environment changes, or state deviations due to external force perturbations. To approach this goal, Receding Horizon Planning (RHP) can be a promising solution. RHP refers to the planning framework that can constantly update the motion plan for immediate execution. To achieve successful RHP, we typically need to consider an extended planning horizon, which consists of an execution horizon that plans the motion to be executed, and a prediction horizon that foresees the future. Although the prediction horizon is never executed, it is important to the success of RHP. This is because the prediction horizon serves as a value function approximation that evaluates the feasibility and the future effort required for accomplishing the given task starting from a chosen robot state. Having such value information can guide the execution horizon toward the states that are beneficial for the future.

Nevertheless, computing such multi-contact motions for a legged robot to traverse uneven terrain can be time-consuming, especially when considering a long planning horizon. The computation complexity typically comes from the simultaneous resolution of the following two sub-problems: 1) selecting a gait pattern that specifies the sequence in which the limbs break and make contact with the environment; 2) synthesizing the contact and motion plan that determines the robot state trajectory along with the contact plan, i.e., contact locations and contact timings. The issue of gait pattern selection introduces combinatorial complexity into the planning problem, while the computation of the contact and motion plan brings high-dimensionality and non-convexity due to the consideration of complex non-linear dynamics constraints.

To facilitate online RHP of multi-contact motions, in this thesis, we focus on exploring novel methods to address these two sub-problems efficiently. To give more detail, we firstly consider the problem of planning contact and motion plans in an online receding horizon fashion. In this case, we pre-specifying the gait pattern as a priori. Although this helps us to avoid the combinatorial complexity, the resulting planning problem is still high-dimensional and non-convex, which can hinder online computation. To improve the computation speed, we propose to simplify the modeling of the value function approximation that is required for guiding the RHP. This leads

to 1) Receding Horizon Planning with Multiple Levels of Model Fidelity, where we compute the prediction horizon with a convex relaxed model; 2) Locally- Guided Receding Horizon Planning—where we propose to learn an oracle to predict local objectives (intermediate goals) for completing a given task, and then we use these local objectives to construct local value functions to guide a short-horizon RHP. We evaluate our methods for planning centroidal trajectories of a humanoid robot walking on moderate slopes as well as large slopes where static stability cannot be maintained. The result of multi-fidelity RHP demonstrates that we can accelerate the computation speed by relaxing the model accuracy in the prediction horizon. However, the relaxation cannot be arbitrary. Furthermore, owing to the shortened planning horizon, we find that locally-guided RHP demonstrates the best computation efficiency (95%-98.6% cycles converge online). This computation advantage enables us to demonstrate online RHP for our real-world humanoid robot Talos walking in dynamic environments that change on-the-fly.

To handle the combinatorial complexity that arises from the gait pattern selection issue, we propose the idea of constructing a map from the task specifications to the gait pattern selections for a given environment model and performance objective (cost). We show that for a 2D half-cheetah model and a quadruped robot, a direct mapping between a given task and an optimal gait pattern can be established. We use supervised learning to capture the structure of this map in the form of gait regions. Furthermore, we also find that the trajectories in each gait region are qualitatively similar. We utilize this property to construct a warm-starting trajectory for each gait region, i.e., the mean of the trajectories discovered in each region. We empirically show that these warm-starting trajectories can improve the computation speed of our trajectory optimization problem up to 60 times when compared with random initial guesses. Moreover, we also conduct experimental trials on the ANYmal robot to validate our method.

ACKNOWLEDGEMENTS

During this research journey, I would like to express my sincere gratitude to everyone who has supported, helped, and advised me.

First and foremost, I would like to profoundly thank my supervisors, Dr. Steve Tonneau and Prof. Sethu Vijayakumar, for giving me excellent academic guidance and personal advice during my time at the Statistical Learning and Motor Control (SLMC) Group. I am also grateful to Dr. Zhibin Li, Dr. Michael Mistry, and Dr. Chris Liu for their insightful feedback and advice during my annual review meetings.

Secondly, I would like to express my utmost thanks to my collaborators for their help during the preparation of our co-authored publications. Specifically, I would like to thank Dr. Sanghyun Kim for building the interface to the robot simulation software, Dr. Teguh Santoso Lembono and Dr. Sylvain Calinon for sharing their insight and expertise on machine learning techniques, Dr. Iordanis Chatzinikolaidis, Dr. Wouter Wolfslag, and Dr. Carlos Mastalli for their discussions and help on improving the quality of manuscripts, Dr. Guiyang Xin for developing a whole-body inverse dynamics controller for the ANYmal robot, Dr. Ke Wang, Dr. Vladimir Ivan, and Jaehyun Shim for helping the development of the robot control stack of our humanoid robot Talos, Dr. Wenqian Du and Dr. Saeid Samadi for their discussions on addressing the challenges encountered in our robot experiment of Talos, as well as their help and patience for taking care of the robot safety during real-world operation of Talos.

In addition to my co-authors, I would like to thank Douglas Howie from the University's workshop, who designed and created a flexible uneven terrain module for testing my locomotion planning algorithms in the real world. Also, I would like to thank PAL Robotics, Dr. Pierre Fernbach and Dr. Olivier Stasse for providing technical and hardware support of the Talos humanoid robot, and Maxime Dufour and Caroline Dulaurent from Artelys for their support on the optimization software Knitro.

Moreover, I would also like to thank my current and former colleagues at the University of Edinburgh, Dr. João Moura, Dr. Daniel Gordon, Dr. Christopher Mower, Dr. Henrique Ferrolho, Dr. Wolfgang Merkt, Dr. Hsiu-Chin Lin, Dr. Agamemnon Krasoulis, Dr. Christian Rauch, Dr. Lei Yan, Dr. Carlo Tiseo, Dr. Xinnuo Xu, Dr. Boyan Gao, Dr. Weihong Li, Dr. Giulia De Togni, Dr. Namiko Saito, Dr. Keyhan Kouhkiloui Babarahmati, Dr. Quentin Rouxel, Dr. Christopher McGreavy, Thomas Corberes, Marina Aoyama, Andreas Christou, Ran Long, Traiko Dinev and Ruaridh Mon-Williams, who have supported me in various of ways during my research. Also,

I would like to thank Euan Morse from the University's administration team, who helped me to achieve my travel to attend the IROS 2022 conference in Kyoto (Japan) during the Covid-19 pandemic. I would also like to thank Aleksandra Zawada and Juliette Sutherland from the University's reception team, who always warm my heart with a smiling face and help me to address various of issues I meet in the building. Moreover, I would also like to thank the cleaning team of the University, who help us to maintain a clean and tidy working space.

Doing a Ph.D. can be stressful. I would like to thank my flatmates, Dr. Theodoros Stouraitis and Dr. Evripidis Gkanias, who are also my colleagues at the University, for their kind support and help during my difficult times. They have not only helped me by giving me technical advice on my Ph.D. project, but also passed me the attitude of dealing with stress and uncertainty caused by research and life.

Additionally, I would like to express my gratitude to my other Greek friends in Edinburgh, Paschalina Oikonomidou, Stergios Giannouloudis, and Despina Kesidou. I enjoyed a lot the time I spent with the Greek community.

Finally, I would like to thank my family for supporting me in pursuing my dream.

PUBLICATIONS

Parts of the research leading to this thesis have previously appeared in the following peer-reviewed publications. Some passages have been quoted verbatim from the respective sources.

JOURNAL ARTICLES

- J. Wang, S. Kim, T.S. Lembono, W. Du, J. Shim, S. Samadi, K. Wang, V. Ivan, S. Calinon, S. Vijayakumar, and S. Tonneau. 'Online Multi-Contact Receding Horizon Planning via Value Function Approximation'. arXiv preprint arXiv:2306.04732, 2023.

  Video: https://youtu.be/oMo_50XIE24

CONFERENCE ARTICLES

- J. Wang, I. Chatzinikolaidis, C. Mastalli, W. Wolfslag, G. Xin, S. Tonneau, and S. Vijayakumar. 'Automatic gait pattern selection for legged robots'. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, USA, 2020.

  Video: https://youtu.be/ylfYnl7sZDg

- J. Wang, S. Kim, S. Vijayakumar, and S. Tonneau. 'Multi-Fidelity Receding Horizon Planning for Multi-Contact Locomotion'. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Munich, Germany, 2021.

  Video: https://youtu.be/csG2uO_foSY

- J. Wang, T.S. Lembono, S. Kim, S. Calinon, S. Vijayakumar, S. Tonneau. 'Learning to Guide Online Multi-Contact Receding Horizon Planning.'. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan, 2022.

  Video: https://youtu.be/oCsOBHHc9XM

DECLARATION

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Edinburgh, United Kingdom, 2023

Jiayi Wang
26th January 2025

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

RHP	Receding Horizon Planning	2
CoM	Center of Mass	11
MIP	Mixed-Integer Programming	20
MINLP	Mixed-Integer Non-Linear Programming	7
TO	Trajectory Optimization	2
EH	Execution Horizon	34
PH	Prediction Horizon	34
MF-RHP	Multi-Fidelity Receding Horizon Planning	35
LG-RHP	Locally-Guided Receding Horizon Planning	31
NLP	Non-Linear Programming	47
MPC	Model Predictive Control	2
DDP	Differential Dynamic Programming	15
k-NN	k-Nearest Neighbor	61
GPR	Gaussian Process Regression	62
NN	Neural Network	62
IMU	Inertial Measurement Unit	101

LIST OF SYMBOLS

m	Robot mass
g	Gravitational acceleration
Ω	Environment model
$\mathcal{S}$	Contact surface
t	Time
T	Total motion duration
τ	Discretized time step interval
μ	Friction coefficient
N_L	Number of limbs
N_{ph}	Number of phases
$l \in \{1, ..., N_L\}$	Limb index
$q \in \{1, ..., N_{ph}\}$	Phase index
$\Lambda \in \{0,1\}^{N_L}$	Contact configuration of the robot
$\boldsymbol{\Lambda} \in \{0,1\}^{\{N_L \times N_{ph}\}}$	Gait pattern matrix
$\boldsymbol{x}$	Robot state vector
$\boldsymbol{u}$	Control input vector
$\boldsymbol{c} \in \mathbb{R}^3$	Center-of-Mass Position
$\boldsymbol{L} \in \mathbb{R}^3$	Angular momentum
$\boldsymbol{\Theta} \in \mathbb{R}^3$	Euler angle of the robot base
$\boldsymbol{\omega} \in \mathbb{R}^3$	Angular velocity vector
$\boldsymbol{p} \in \mathbb{R}^3$	Contact location vector
$\boldsymbol{f} \in \mathbb{R}^3$	Contact force vector

To a dreamland where robots can help humans.

1

INTRODUCTION

1.1 SCOPE

In recent years, we have witnessed rapid growth in the development of mobile robots. Especially, the advances in wheeled systems have enabled several real-world applications. For instance, we are seeing wheeled robots getting into household environments to vacuum our floors; restaurants and supermarkets are deploying wheeled platforms to handle delivery tasks; autonomous driving cars are getting matured in controlling the steering of their wheels. Despite these successful applications, wheeled robots are often constrained in structured environments, i.e, flat floors, and urban environments built with roads.

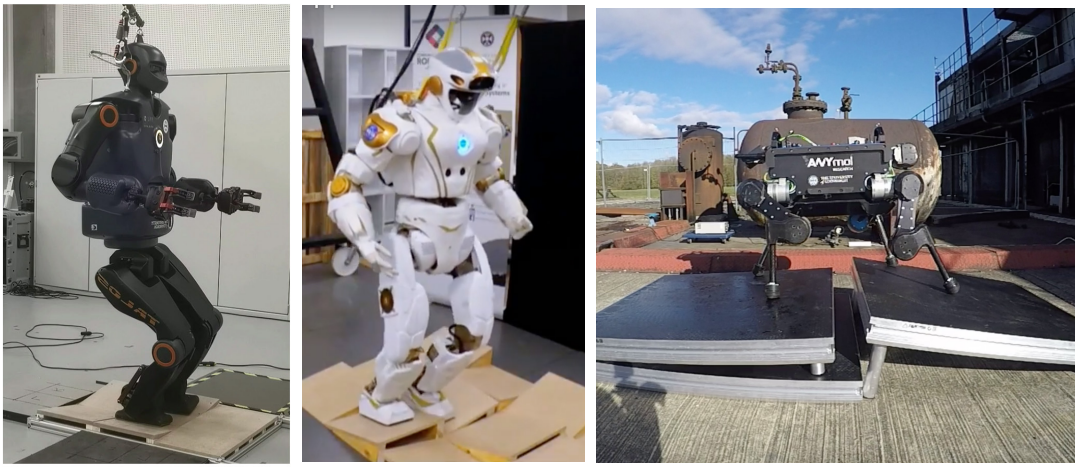


Figure 1: Legged robots can traverse uneven terrain by utilizing multiple contacts between their end-effectors and the environment. The picture of Valkyrie (middle) is taken from the video https://youtu.be/MoHv3bRfdok.

On the other hand, another type of mobile robot—legged systems (humanoids and quadrupeds)—has shown superior mobility than the wheeled systems. For instance, legged robots can overcome terrain irregularities such as slopes, gaps, and stairs. This is achieved by utilizing multiple contacts between their end-effectors and the environment (see examples in Fig. 1). This capability makes legged robots particularly

useful for tasks taking place in unstructured environments, i.e., exploring disaster zones, which can be dangerous for humans [4].

In order to navigate in arbitrary environments, legged robots necessarily require the ability to automatically plan their motions to traverse uneven terrain. Nevertheless, planning such complex movements is often a challenging task. This planning problem—also known as multi-contact motion planning—not only requires the computation of a robot state trajectory, but also needs to coordinate a sequence of contact events, i.e., when and where a limb should create a contact [11, 81, 94, 109]. In the meantime, this planning problem also needs to consider complex non-linear dynamics constraints to govern the motion. These challenges together render the multi-contact motion planning problem combinatorial, high-dimensional, and non-convex [97].

Due to these computational challenges, traditional legged robot control frameworks usually pre-compute the multi-contact motion plans offline and then track the planned trajectories with a controller [94, 109]. However, when deploying legged robots in the real world, robots can encounter disturbances such as environment changes and state deviations. These unexpected changes can cause the pre-planned motions to become invalid, and continue executing these pre-planned motions can lead the robot to fall [5, 34, 54].

To facilitate reliable operation in the real world, it is essential that legged robots can have the capability to online (re)-plan their motions. To approach this goal, Receding Horizon Planning (RHP) [68, 77] is a promising solution. Borrowing the idea from Model Predictive Control (MPC) [24, 28, 62, 73], RHP aims at constantly updating the motion plan for immediate execution based on the state of the robot as well as the latest observation of the environment (see Fig. 2). Such a re-planning strategy can enable legged robots to generate adaptive behaviors in response to changing conditions, i.e., dynamics changes of the environment and state deviations.

Typically, RHP frameworks compute the motion plan by solving a finite-horizon Trajectory Optimization (TO) problem [8], where the planning horizon usually consists of two parts: 1) an execution horizon that plans the optimal actions for execution; and 2) a prediction horizon that foresees the future beyond the execution horizon (see an example in Fig. 2). Although RHP frameworks never execute the prediction horizon, it plays a vital role in the success of RHP. Viewing the RHP problem from Bellman's perspective [7], the prediction horizon can be seen as an approximation of the value function—which predicts the feasibility and the future effort for achieving the task starting from a given robot state. Such a value function can guide the execution horizon toward optimal actions that are beneficial for the future and avoid short-sighted decisions, e.g., causing the robot to fall or get stuck in the local minima of the environment.

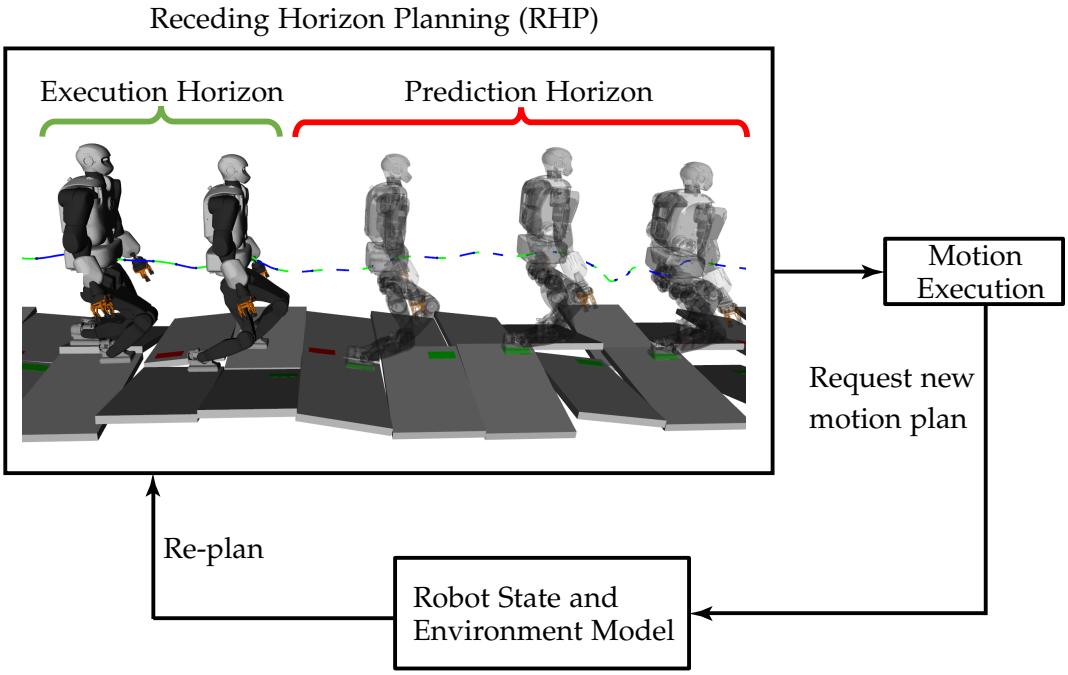


Figure 2: The core idea of Receding Horizon Planning (RHP). RHP frameworks is designed to constantly update the motion plan for immediate execution based on the state of the robot and the latest observation of the environment. This planning strategy is particularly useful for dealing with changing conditions, i.e., environment changes and state drifts. In RHP, the planning horizon usually needs to include: i) an execution horizon that plans the motion for immediate execution, ii) a prediction horizon (not executed) that foresees the future. Considering the prediction horizon is important, as it guides the execution horizon by predicting whether the motion to be executed can benefit the future operation of the robot or not.

Nevertheless, planning multi-contact motions over an extended horizon typically results in expensive computation. This computational challenge usually prevents us from achieving fast multi-contact motion planning in an online receding horizon fashion. Throughout this thesis, we will focus on exploring novel methods to accelerate the computation speed of multi-contact motion planning.

1.2 PROBLEM STATEMENT

We consider the problem of planning multi-contact motions for a legged robot to traverse uneven terrain. Particularly, we are interested in finding efficient planning methods that can allow *online RHP of multi-contact motions*.

However, computing multi-contact motion plans in an online setting is a challenging task. As mentioned in Section 1.1, this planning problem is combinatorial, high-dimensional, and non-convex. To give more detail, these computation challenges of multi-contact motion planning mainly result from the joint resolution of the following two sub-problems (see Fig. 3):

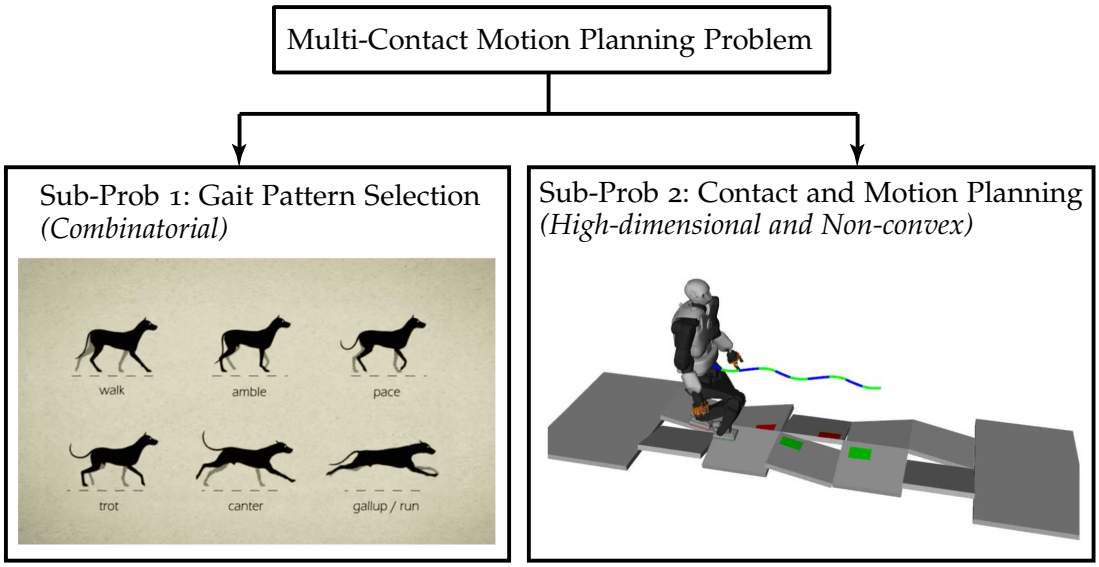


Figure 3: Multi-contact motions planning usually requires the simultaneous resolution of the following two sub-problems: 1) gait pattern selection which decides the sequence in which the limbs break and make contact with the ground, 2) contact and motion planning, which computes the robot state trajectory along with the contact plan (contact locations and timings). In the same time, contact and motion planning also requires the consideration of complex system dynamics constraints. The picture illustrating the gait pattern selection problem is taken from the video https://www.youtube.com/watch?v=PVvZKcKBTtg.

- *Sub-Problem 1: Gait Pattern Selection (Combinatorial)*

  When planning multi-contact motions, an important issue is to make a discrete choice on the gait pattern—the sequence in which the limbs break and make contact with the ground. Such a discrete decision typically has a large impact on the feasibility and optimality of the motion [83, 88, 113]. This is because the gait pattern determines the order of contact activation, which characterizes the dynamics and kinematics of the robot. However, finding an appropriate gait pattern is a discrete search problem, which brings combinatorial complexity into the problem.

- *Sub-Problem 2: Contact and Motion Planning (High-dimensional and Non-convex)*

  In addition to selecting the gait pattern, multi-contact motion planning also requires contact and motion planning [11], which refers to the problem of finding a feasible state trajectory along with a sequence of contacts (locations and timings) that comply with the chosen gait pattern. In the meantime, the decisions of these variables cannot be arbitrary and have to respect the complex nonlinear system dynamics constraint that governs the motion, i.e., centroidal dynamics [16, 75] or whole-body dynamics [62]. However, computing contact and motion plans while considering complex dynamics typically introduces high-dimensionality and non-convexity into the problem.

To achieve online RHP of multi-contact motions, it is critical that we can find computationally efficient methods to address these two sub-problems.

To approach this end, we start from the case where we pre-define the gait pattern as a priori (an ad-hoc solution of sub-problem 1), and then we focus on addressing the contact and motion planning problem (sub-problem 2) in an online receding horizon fashion. Although in this case, we avoid the combinatorial search of the gait pattern, the resultant contact and motion planning problem is still high-dimensional and non-convex, which imposes challenges for achieving online RHP. To improve the computation speed, we propose to find simplifications to the modeling of this planning problem. Particularly, we explore the possibility to reduce the computation complexity introduced by the value function approximation that is required for guiding the RHP, i.e., computing the prediction horizon with simplified models or learning an approximated value function model from past experience. The details of our approach and the experiment result are presented from Chapter 4 to Chapter 8.

For the issue of selecting the gait pattern (sub-problem 1), we conduct a computational study offline to investigate how the task specification and the environment model can influence the choice of the gait pattern. Throughout this empirical study, we find that there exists a map from the task conditions to the gait pattern selection. We can use this map to guide the gait pattern selection during online planning, i.e., selecting the gait pattern by querying the map. We describe the approach and result of this empirical study in Chapter 9. Next, we describe our contributions in detail.

In this section, we summarize the main contributions of this thesis. These contributions are categorized into two groups that focus on addressing the two sub-problems involved in the multi-contact motion planning (summarized in Section 1.2 and Fig. 3):

1. ***Online Receding Horizon Planning of Contact and Motion Plans via Value Function Approximation***

The first part of our contributions focuses on achieving online RHP of contact and motion plans on uneven terrain (sub-problem 2 as described in Section 1.2). In this case, we pre-define the gait pattern as a priori, and we aim to compute the state trajectory along with contact locations and timings in an online receding horizon fashion.

However, computing such complex motion plans in an online receding horizon fashion is challenging. We recall that RHP necessarily requires the consideration of a value function that is used to direct the execution horizon (plans the motion being executed) towards a state that is favorable for the future. Traditionally, the value function is approximated by computing trajectories in a prediction horizon (not executed) that foresees the future. However, approximating the value function with the prediction horizon can significantly increase the computational complexity, especially when planning multi-contact motions where complex nonlinear dynamics constraints need to be considered.

To increase the computation speed, we propose to reduce the computation complexity by finding computationally efficient approximations of the value function. This leads to:

- **Receding Horizon Planning with Multiple Levels of Model Fidelity**, where we follow the trajectory-based formalism that approximates the value function by computing trajectories in the prediction horizon. However, instead of considering an accurate dynamics model (usually non-convex), we propose to plan the prediction horizon with convex relaxed models. This allows us to reduce the overall computation complexity of the problem and facilitates online computation. We explore and compare three multi-fidelity RHP candidates with different convex relaxations employed in the prediction horizon.

- **Locally-Guided Receding Horizon Planning**, where we approximate the value function with a learned oracle. This oracle is designed to predict local objectives

as intermediate goal states for completing a given task while considering the environment model around the robot. We use these local objectives to build local value functions for guiding a short-horizon RHP to plan the execution horizon. To obtain the oracle, we take a supervised learning approach, where we train the oracle from the dataset offline generated by the traditional RHP approach which plans the execution horizon as well as the prediction horizon with an accurate dynamics model.

We carry out extensive evaluations and analysis on the computation performance of multi-fidelity RHP and locally-guided RHP in the context of planning centroidal trajectories of the humanoid robot Talos [90] walking on uneven terrain, and we use a whole-body inverse dynamics controller [27] to validate the dynamic feasibility of the planned trajectories in simulation. Our results show that locally-guided RHP achieves the best computation efficiency (see our experiment results in Table 2 and Table 5). This computation advantage enables us to demonstrate online receding horizon planning of our real-world humanoid robot Talos walking in dynamic environments that change on-the-fly. These contributions are also presented in our previous publications [102–104].

2. ***Automatic Gait Pattern Selection for Legged Robots***

In this part of the contribution, we focus on addressing the problem of selecting a gait pattern (sub-problem 1 as described in Section 1.2). As mentioned in Section 1.2, the choice of the gait pattern can have a large impact on the feasibility and optimality of a motion with respect to a task. However, finding an appropriate gait pattern introduces combinatorial complexity into the multi-contact motion planning problem, which prevents online computation. To address this issue, we propose the idea of building a map from task specifications to the gait pattern selection for a given environment model and performance objective (cost). By querying the map, we can quickly decide the gait pattern during online planning.

To generate the map, we use Mixed-Integer Non-Linear Programming (MINLP) to compute offline a dataset of optimal gait patterns for a given set of task specifications and environments. Then, we employ a neural network to encode the map. We use the term "optimal" to refer to the locally optimal solutions enhanced by multiple restarts. We establish optimal gait pattern selection maps for a 2D half-cheetah model and the 3D quadruped robot ANYmal. These maps typically form several contiguous regions for which a particular gait pattern is optimal. Furthermore, we also observed that the optimal trajectories within each gait region are qualitatively similar. We find the

mean of these trajectories can be used to warm-start our optimization-based planning algorithm. For example, when computing the motion plan for a chosen gait pattern, we can use the mean of the discovered trajectories as a seed to initialize the optimization solver. We empirically show that using the warm-starting trajectories can further improve the convergence speed up to 60 times faster when compared with random initial seeds. Lastly, we also conducted experimental trials on the ANYmal robot to validate the dynamic feasibility of the motion discovered by our MINLP approach. These contributions are also presented in our previous publication [101].

1.4 THESIS OUTLINE

We organize the thesis as follows:

- Chapter 1 gives an introduction and overview of the thesis.
- Chapter 2 reviews the existing literature for planning multi-contact motions.
- Chapter 3 provides a formal description of the multi-contact motion planning problem and an analysis of the associated computational challenges.
- Chapter 4 presents a general description of the RHP problem for computing contact and motion plans, along with the core idea of our proposed value function approximation techniques for accelerating the computation speed. Furthermore, we also provide the detailed formulation of the traditional RHP approach (baseline) in the context of finding contact and motion plans.
- Chapter 5 presents the technical detail of our multi-fidelity RHP approach.
- Chapter 6 describes the implementation detail of our locally-guided RHP approach.
- Chapter 7 presents a simulation study to evaluate the computation performance of multi-fidelity RHP and locally-guided RHP in various multi-contact scenarios.
- Chapter 8 describes the real-world experiments that demonstrate online RHP on our humanoid robot Talos.

- Chapter 9 presents our approach to achieve automatic gait pattern selection during online planning. The core idea is to leverage offline computation to establish gait pattern selection maps that can be queried online.

- Chapter 10 summarizes our contributions and provides an outlook of future research directions.

2

LITERATURE REVIEW

In the past decades, the research community has attempted various methods to address the multi-contact motion planning problem. In this chapter, we provide an overview of these approaches. More specifically, in Section 2.1, we first revisit the methods that are designed for solving the contact and motion planning problem, which focuses on finding the state trajectory and the contact plan of a legged robot traversing uneven terrain with a pre-defined gait pattern (sub-problem 2 as described in Section 1.2). Then, in Section 2.2, we review the approaches that could optimize the gait pattern (sub-problem 1 as described in Section 1.2).

2.1 CONTACT AND MOTION PLANNING

In this section, we present a literature review on the approaches that deal with the contact and motion planning problem. We recall that the term "contact and motion planning" refers to the issue of computing a feasible state trajectory and a sequence of contacts (locations and timings) for a legged robot to traverse uneven terrain under the assumption of following a given gait pattern (see an example in Fig. 4).

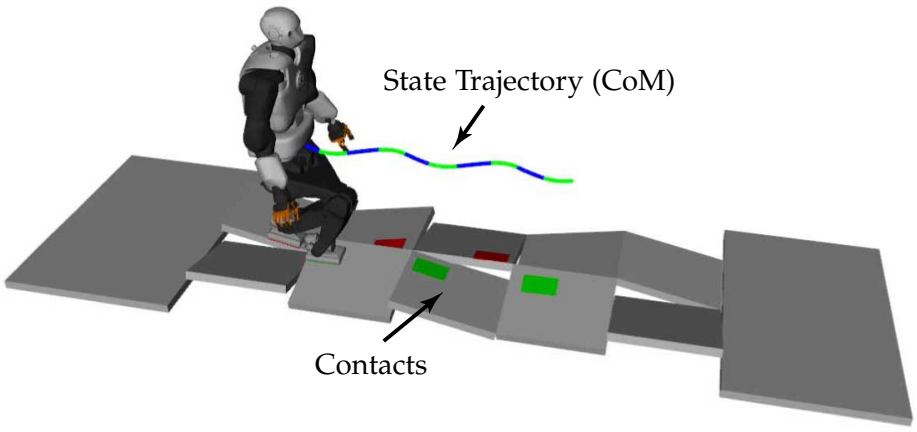


Figure 4: Contact and motion planning refers to the problem of finding a feasible state trajectory, e.g., the Center of Mass (CoM) trajectory (the blue-green curve), along with a sequence of contacts (the red and green rectangles), that can allow a legged robot to traverse uneven terrain. Usually, the planning assumes a pre-defined gait pattern (the order of contact activation), i.e., the bipedal walking gait that creates contact with the left and right feet in a cyclic fashion. This picture is generated by the visualization software called Gepetto-viewer [55].

To address this planning problem, one of the options is to use divide-and-conquer methods, which solve the planning problem by dividing it into a sequence of sub-problems with smaller sizes (Section 2.1.1). Alternatively, another option for computing contact and motion plans is to solve a Trajectory Optimization (TO) problem, which jointly optimizes the state trajectories along with the contact locations and timings (Section 2.1.2). In this section, we provide a thorough overview of these two families of planning methods. Additionally, we also present a review on using learning techniques to accelerate the computation of the contact and motion planning problems (Section 2.1.3).

2.1.1 *Divide-and-Conquer Methods*

Computing contact and motion plans for a legged robot to traverse uneven terrain can be a challenging task. Although the gait pattern is often fixed as a priori, solving this planning problem is still a complex issue, as it requires simultaneous resolution of a feasible state trajectory, along with a sequence of associated contact states (locations and timings) to achieve that trajectory [11]. To tackle such complex planning problems, one of the options is to employ the divide-and-conquer strategy, which splits the contact and motion planning problem into a sequence of sub-problems that are more tractable to solve (Fig. 5). Typical examples of using divide-and-conquer methods to address the contact and motion planning problem can be found in [6, 32, 33, 38, 47, 52, 82, 94, 108, 110].

Generally speaking, the computation pipeline of these divide-and-conquer methods usually consists of the following three stages (see an illustration in Fig. 5):

1. *Stage 1—Guide path planning*

   In the first stage, the divide-and-conquer methods focus on finding a guide path for the root of the robot (base or torso) to navigate through the given environment, while moving towards the desired goal. To increase the chance of finding a feasible guide path that can be achieved by the robot, divide-and-conquer methods usually need to incorporate heuristics to examine the feasibility constraints (robot dynamics and kinematics) at an approximate level. For instance, [47, 52, 82, 108, 110] engineer a terrain cost-map to reflect the traversability (roughness) of each terrain region (see an example in Fig. 6-a). This terrain cost-map can be used to drive the guide path to avoid challenging terrain regions that may cause failures, i.e., large slopes and gaps scored with high cost values. Alternatively, [94] proposes to impose a reachability condition

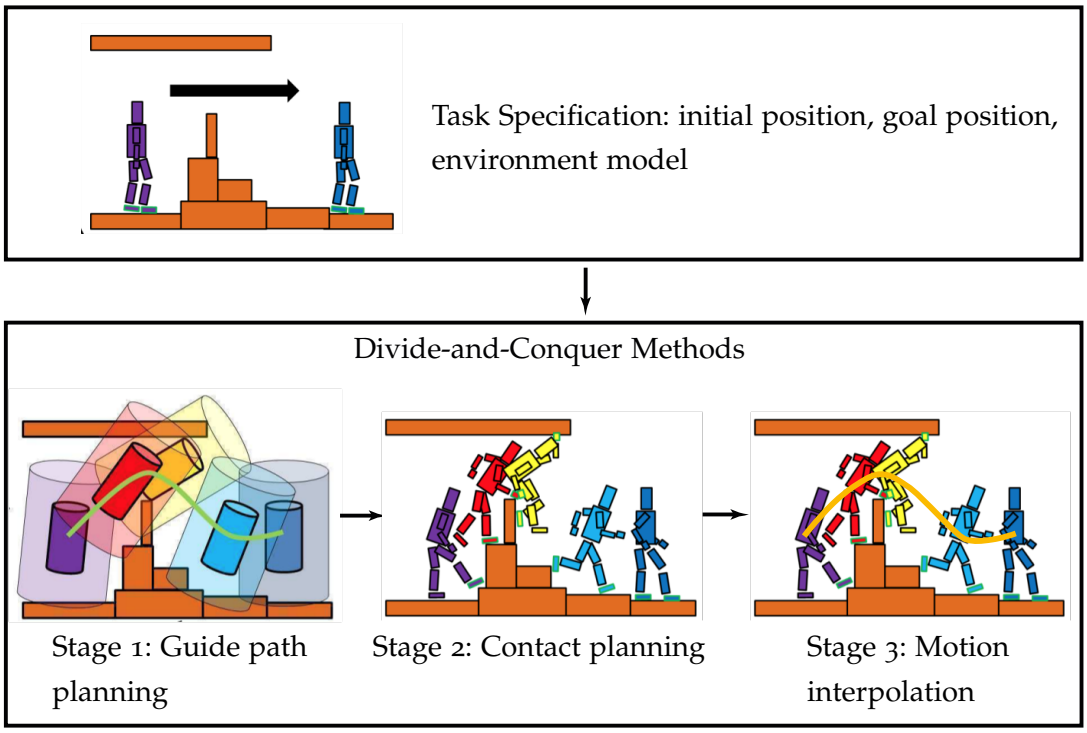


Figure 5: Divide-and-conquer methods solve the contact and motion planning problem by decomposing it into a sequence of sub-problems that are more tractable to compute. Typically, the computation pipeline consists of the following three stages: i) Stage 1—guide path planning, which computes an approximate path for the root of the robot (base or torso) to go across the given terrain; ii) Stage 2—contact planning, which generates a sequence of contact postures along the guide path, i.e., determining the locations of each active contact; iii) Stage 3—motion interpolation, which interpolates the motion between the planned contact postures. This figure is modified from [93].

during the guide path planning (see an example in Fig. 6-b). This reachability condition describes the important features that a feasible guide path should have: i) the workspace of the robot end-effectors must have intersections with the environment to allow contact creation, and ii) the root of the robot (base or torso) should avoid collisions with the environment.

2. *Stage 2—Contact planning*

   In the second stage, the task of the divide-and-conquer methods is to find a contact plan along the guide path, i.e., a sequence of footsteps. Similar to the first stage, heuristics are often involved in guiding contact planning. For example, [6, 33, 47, 52, 82, 108, 110] design reward terms to guide discrete search methods to choose contact locations with desired properties, i.e., stepping on near-flat surfaces to minimize the risk of foot slippage, being away from robot

kinematics limit. Similarly, [94] uses a sampling-based approach to search for contact locations. Particularly, they are interested in finding contact locations that can allow the robot to establish standing postures in static equilibrium. This is achieved by using Linear Programming (LP) to check the validity of sampled robot configurations.

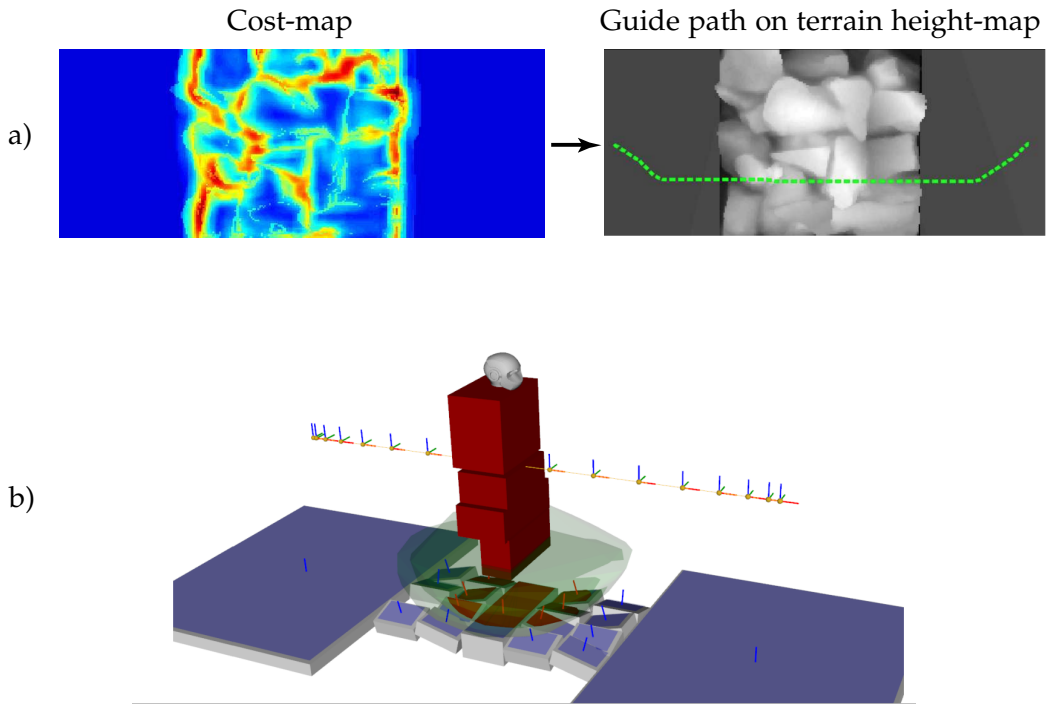


Figure 6: Heuristics that are used to direct the guide path planning in divide-and-conquer methods: a) Computing the guide path with a terrain cost-map. The cost-map is modeled as a 2D grid, where each point has an associated cost that reflects the roughness of the local terrain around that point. The cost is computed as a linear combination of several terrain features of the local terrain, e.g., standard deviation of terrain heights, average slope of the terrain. This cost-map can direct the planner to choose a guide path (the green line) that tends to move into the regions with smaller terrain elevation and inclination changes (blue regions), while avoiding challenging regions, i.e., large slopes and gaps (red regions). The picture is adapted from [52]. b) Generating the guide path based on a reachability condition, which constrains the reachable workspace of robot limbs (green spheres) to intersect with the environment to allow contact creation, while in the meantime avoiding collisions between the root of the robot and the environment (red boxes). The picture is adapted from [96].

3. *Stage 3—Motion interpolation*

In the last stage, the divide-and-conquer computes the state trajectory based on the contact plan generated in the previous stage. This state trajectory can be a base trajectory that satisfies the static stability [52, 82, 108], or a CoM trajectory that can achieve dynamic locomotion behaviors

Owing to the decomposition of the contact and motion planning problem, the divide-and-conquer methods can achieve fast computation of the motion plan [6, 33, 35, 94, 110]. However, this decomposition also brings a *feasibility issue*: each sub-problem must be addressed in the feasibility domain of the subsequent sub-problems [36]. For instance, in the first stage, we should ensure that it is always possible to find a sequence of contacts to achieve the planned guide path [18]. Similarly, in the second stage, we should guarantee that there always exists a feasible state trajectory that can follow the planned contact sequence. Unfortunately, in divide-and-conquer methods, these feasibility constraints are evaluated at an approximated level, which can introduce the risk of failing to find a solution [18, 35, 36]. To alleviate this issue, researchers have proposed methods to tighten the feasibility constraints in each stage, i.e., using reinforcement learning to learn a guide path planning policy [18], using convex optimization to examine whether a candidate contact transition (re-allocating a contact) is dynamically feasible or not [35].

2.1.2 *Trajectory Optimization (TO) Methods*

Apart from the divide-and-conquer methods, another option for synthesizing contact and motion plans is to solve a TO problem [8] that jointly optimizes the motion of the robot, along with the contact plan, i.e., a sequence of contact locations and timings (see an illustration in Fig. 7). Compared to divide-and-conquer methods, TO-based methods can easily ensure the feasibility of the motion plan by imposing constraints in the TO formulation. Typically, these TO problems can be solved by using either direct collocation [8, 49] or Differential Dynamic Programming (DDP) [62].

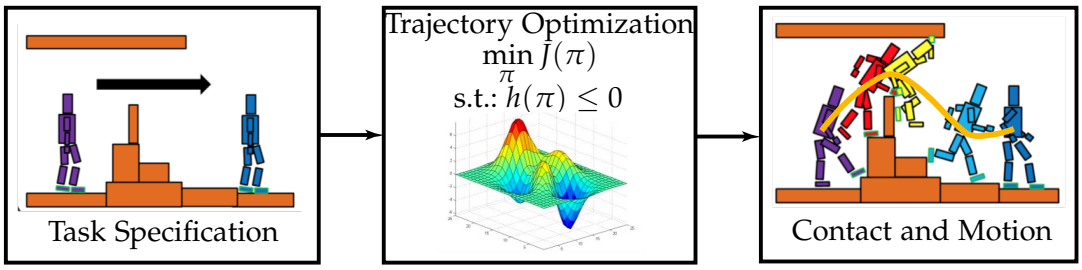


Figure 7: The TO methods computes the contact and motion plan (denoted as π) by solving an optimization problem that can jointly optimize the state trajectory and the contacts (locations and timings). The dynamic feasibility of the motion plan can be easily ensured by imposing dynamics constraint on the TO formulation, i.e., the constraint h. The pictures are modified from [93].

Nevertheless, computing contact and motion plans with TO can be a time-consuming process. This computational challenge mainly comes from the consideration of the complex dynamics constraint that governs the motion (usually high-dimensional and non-convex).

To give more detail, computing contact and motion plans for a legged robot to traverse challenging terrains necessarily requires the consideration of the whole-body dynamics of the robot (see the left figure in Fig. 8). The whole-body dynamics model considers the mass and inertia of every link and relates the joint torques to the base and joint accelerations. In the past, TO-based methods have demonstrated impressive motions using the whole-body dynamics model [31, 37, 51, 53, 61, 85, 92]. However, due to the high-dimensionality and non-convexity of the whole-body dynamics model, these approaches often struggle to achieve online computation unless we pre-define the contact timings and locations as a priori [62, 66, 67].

Alternatively, we can also compute the contact and motion plans using the centroidal dynamics model [16, 75] (see the right figure in Fig. 8). Compared to the whole-body dynamics model, the centroidal dynamic model has lower dimensionality, as it only considers the dynamics of the total linear and angular momenta expressed at the CoM. Moreover, approximations can be introduced on the robot kinematics and the momentum variation that results from the motions of each individual link. Although these approximations can lead to failures to achieve a corresponding whole-body motion, the centroidal dynamics model is widely used for generating contact and motion plans due to its reduced dimensionality [16, 17, 23, 35, 40, 41, 105]. Unfortunately, the centroidal dynamics model is still non-convex, except when strong limiting assumptions are made, e.g., pre-defined gait, flat/co-planar surfaces, fixed CoM height [30, 45, 107]. The non-convexity of the centroidal dynamics model typically stems from the bi-linear terms resulting from the cross-product operation involved in the angular dynamics. Such non-convexity often prevents the TO from achieving online computation.

To improve the computation efficiency of TO-based planning methods, researchers have explored the possibility to further reduce the complexity of the centroidal dynamics model. In general, these approaches focus on finding convex approximations of the centroidal dynamics model.

For instance, [14, 15, 35] propose convex inner approximation approach that searches for a solution within a subset of all possible trajectories. Although convex inner approximation methods can achieve fast computation, these methods may fail to find a solution due to the reduced search space [35]. Furthermore, these convex inner approximation approaches usually assume pre-defined contact locations and timings, which can limit the range of the motions that can be achieved.

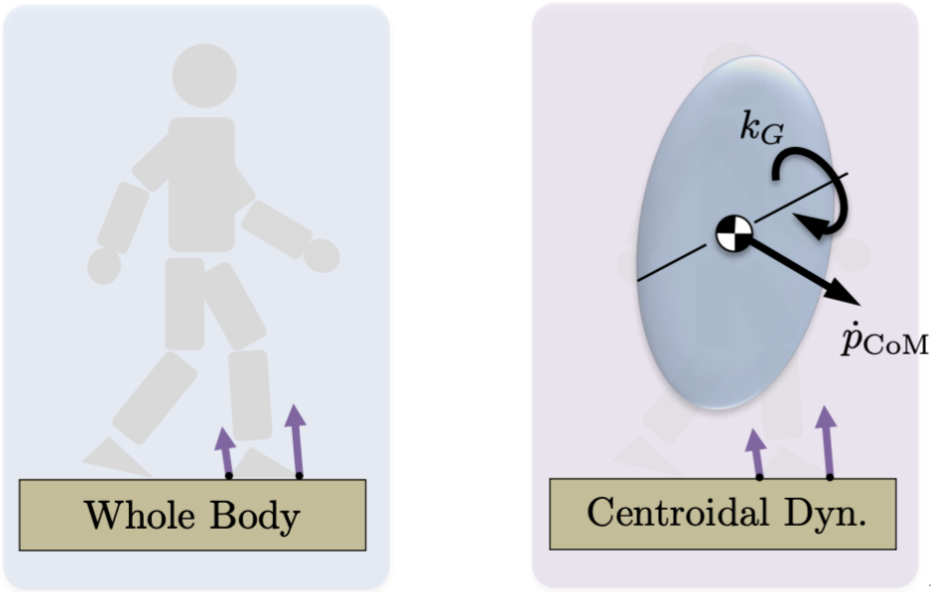


Figure 8: The whole-body dynamics model (left) considers the mass and inertia of every link, i.e., the articulated structure shown in the left picture. In contrast, the centroidal dynamics model (right) is low-dimensional as it only considers the dynamics of the total linear and angular momenta expressed at the CoM, where $\dot{p}_{CoM}$ and k_G denote the linear and angular momentum of the CoM, respectively. In both pictures, the purple arrow represents the contact force exerted by the robot foot. The figures are adapted from [106].

Alternatively, [2, 22, 80] propose convex outer approximation methods that introduce convex relaxations into the centroidal dynamics model. Although the model complexity is reduced, the convex outer approximation may generate motions that violate the system dynamics constraints and cause tracking failures. Although it is possible to gradually tighten the convex relaxation using iterative schemes, this requires the design of customized solvers [80].

In this thesis, we propose a multi-fidelity TO formulation tailored for achieving online RHP of contact and motion plans. More specifically, in our formulation, we employ an accurate model in the execution horizon (to be executed in each cycle), while using a relaxed model to plan the prediction horizon (not executed). Such a combination ensures the motion to be executed is always dynamically consistent, while at the same time reducing the overall computation complexity of the TO problem. Also, this formulation is straightforward to implement and can be interfaced directly with off-the-shelf solvers.

A similar approach to the one we propose was introduced recently by [59]. The authors present a MPC framework based on DDP that combines whole-body dynamics

and a non-convex model with reduced order (single-rigid body model [105, 109]) in a single formulation. Successful MPC of 2D quadrupedal locomotion and humanoid running has been demonstrated on flat surfaces. In contrast, our thesis emphasizes achieving online RHP of centroidal trajectories for uneven-terrain locomotion. This problem requires a careful selection of contact locations and timings to modulate the centroidal momenta. In this regard, the simplified model employed in the prediction horizon needs to be carefully designed, as the quality of the model can significantly affect the performance of the framework. Furthermore, instead of searching for non-convex models with reduced order, we focus on finding appropriate convex relaxations for the prediction horizon.

2.1.3 *Learning to Accelerate Contact and Motion Planning*

To accelerate the computation speed of contact and motion planning, researchers also explored the usage of machine learning techniques to alleviate the computation burden of the planning problem. For instance, when using discrete search algorithms to search for contact and motion plans, an important issue is to examine the dynamic feasibility of each candidate contact transition (creating or breaking a contact). Although we can use TO methods to evaluate whether a contact transition is feasible or not, this usually leads to heavy computation, especially when there are many candidate contact transitions need to be examined. To address this issue, [60] proposes to use supervised learning to learn a classifier to tell the feasibility of a given contact transition. This learned model is then used to help the A* algorithm to find dynamically feasible contact and motion plans efficiently.

Alternatively, machine learning techniques can also be used to bootstrap the TO-based methods for computing contact and motion plans. For example, [24, 57, 68] propose to learn near-optimal solutions to warm-start the TO solvers. The core idea behind these approaches is to initialize the TO solver to a place that is close to a feasible local minimum. This can lead the TO solver to converge with fewer iterations and achieve online computation.

Furthermore, another strategy for improving the computation efficiency of TO is to shorten its planning horizon. This can be achieved by using a learned value function model placed as the terminal cost of the TO formulation [25, 76, 100, 118]. In this thesis, we follow this idea and propose locally-guided RHP, which focuses on learning a value function model for computing contact and motion plans on uneven terrain. However, learning a value function for the contact and motion planning problem can be challenging. The main difficulty is that the value function is defined in a coupled state-environment space, which requires a flexible parameterization that can

capture the landscape changes of the value function with respect to environment variations [25]. To deal with this issue, we propose to learn an oracle to predict intermediate goal states for completing a given task based on the current state, the final goal, and the environment, and then we construct local value functions based on these intermediate goal states.

From a broader point of view, the oracle can be seen as a sequential action predictor that is designed to continuously predict immediate actions for accomplishing a given task. However, when using a learned model to predict sequential actions, the prediction accuracy can decrease dramatically once the robot reaches a state that is unexplored in the training dataset. This problem is known as *distribution shift* [84], which can be mitigated by data augmentation, i.e. adding demonstrations from the states that either appeared from the roll-out of the learned policy [84, 99], or sampled from the expert policy with injected noise [56]. In this thesis, we present a similar but more targeted data augmentation strategy which focuses on demonstrating corrective actions from the states that cause convergence failures, and we demonstrate this approach can improve the prediction accuracy of the oracle.

2.2 OPTIMIZING THE GAIT PATTERN

In the previous section, we revisit the methods that are designed to address the contact and motion planning problem. Typically, these methods compute the motion plans based on a given gait pattern—the contact sequence in which the limbs break and make contact with the ground. Fixing the gait pattern can allow us to avoid dealing with combinatorial complexity. However, this can also limit the range of achievable motions, which can affect the feasibility and optimality of the motion with respect to a given task. To allow automatic reasoning of the gait pattern selection, researchers have proposed a couple of techniques. In this section, we will present an overview of these approaches.

Recently, the advances in TO have enabled us to explore the modulation of gait patterns through continuous optimization processes. These formulations owe their success to techniques such as i) enforcing complementary conditions [64, 81], ii) smoothing of the discontinuous contact events [70, 72, 109], iii) phase-based parameterization of individual limbs [70, 109], and iv) resolving system dynamics through contacts in the inner loop of a bi-level optimization framework [13, 92]. Generally speaking, these methods typically require to smooth the discontinuous contact dynamics, such that they can numerically compute a local search direction (the gradients of the problem) between the motions characterized by different gait patterns or contact sequences. However, Toussaint et al. [97, 98] argue that, when using continuous optimization

techniques to optimize the gait pattern or the contact sequence, the landscape of this optimization problem is highly non-convex, i.e., different gait patterns or contact sequences imply different local minima. Using the gradient itself is usually insufficient to allow the exploration of a wide range of possible gait patterns or contact sequences, as the optimizer can get stuck in a local minimum. In the worst case, the optimizer can be trapped in an infeasible local minimum and fail to generate valid motions. Furthermore, none of these approaches has shown the capability to achieve online computation.

Alternatively, a more systematic way to search for the gait pattern is to embrace the combinatorial complexity: evaluate multi-contact motion plans with different gait patterns and then select the feasible ones or the optimal one. The naive form of this approach is exhaustive search. To achieve better search efficiency of the gait pattern, we can use Mixed-Integer Programming (MIP) techniques. The MIP methods improve the search efficiency by using Branch-and-Bound algorithm [20], which performs a tree search among the integer variables that are used to model the gait patterns or the contact sequences [2, 26, 79]. This Branch-and-Bound algorithm can increase search efficiency, as it keeps pruning branches that may have high costs and restricting the searches to the promising ones.

To further improve the computation speed with MIP, Aceituno-Cabezas et al. [2] introduce the following assumptions: i) use a single binary value to represent the entire foot-swing phase, ii) employ a convex approximation of the centroidal dynamics model, and iii) fix the phase duration. These assumptions allow Aceituno-Cabezas et al. [2] to formulate the gait pattern selection problem as a Mixed-inter Convex Programming Problem, which can be solved with efficient mixed-integer convex optimization algorithms [39]. However, these assumptions come with the cost of having a limited range of achievable motions. For instance, the first assumption cannot capture unsynchronized breaking and making of contacts, and thus it can only model symmetrical gait patterns, e.g., walking and trotting. The second assumption does not properly encode the typical changes of the angular momentum needed in highly dynamic motions, i.e., bounding. The last assumption—fixing phase duration—restricts the search space of the possible trajectories being considered. In this thesis, we present a MINLP approach for optimizing the gait pattern. Our approach follows the same paradigm as in [2], but our modeling can allow both symmetric (walking, trotting) and asymmetric gait patterns (galloping). Additionally, the angular dynamics is accurately described by a full nonlinear model, and the phase durations are considered as decision variables. These changes allow us to consider a wide range of motions.

Generally speaking, due to the combinatorial complexity, MIP methods are usually computationally expensive, especially when there involves a large number of integer

variables. Such computation bottleneck often prohibits online usage. To achieve online planning of multi-contact motions, in this thesis, we explore the possibility to extract a gait pattern selection policy from the MINLP solutions computed offline. Then we use supervised learning to learn a direct map from task specifications to the optimal gait patterns for different performance objectives and environmental models. By querying this gait pattern selection map, we can quickly decide the gait pattern during online planning.

2.3 CONCLUSION

In this chapter, we presented a literature review on existing methods designed for computing multi-contact motion plans. More specifically, we firstly reviewed the methods designed for computing contact and motion plans while following a pre-defined gait pattern. This involves two kinds of methods: 1) divide-and-conquer approach, which decomposes the contact and motion planning problem into a sequence of sub-problems; 2) TO-based approaches, which optimize the robot motion and the contact plan simultaneously. The divide-and-conquer methods can achieve fast computation owing to their decomposed structure. However, this decomposition strategy also introduces a feasibility issue: the solution of each sub-problem will limit the search space of the subsequent sub-problems, and this can introduce the risk of failing to find a motion plan, if the search space of any sub-problems is too restricted. In contrast, the TO-based methods jointly optimize all the decision variables while respecting the dynamics constraints imposed on them. Such a holistic optimization approach avoids the feasibility issue arising from the divide-and-conquer methods. However, given the high-dimensionality and non-convexity of the dynamics constraints, TO-based methods often struggle to achieve online computation. Additionally, we also presented a review of the recent attempts to accelerate the computation speed of contact and motion planning with machine learning techniques. In this thesis, building upon previous work, we focus on further improving the computation efficiency of TO-based methods, such that we can achieve online RHP of contact and motion plans. To achieve this goal, we explore the possibility to reduce the computation complexity of the TO problem. This includes: 1) finding simplifications to the modeling of the TO; 2) shortening the planning horizon with a learned value function model.

Furthermore, in this chapter, we also reviewed the methods that can optimize the gait pattern. These methods include: 1) TO-based continuous optimization methods, and 2) MIP-based discrete optimization methods. Nevertheless, researchers argue that the continuous optimization methods may not be able to explore a wide range of motions with different gait patterns, as they are local search methods and can get

stuck in a local minimum. Furthermore, these continuous optimization approaches can still take a considerable amount of time to compute. In contrast, MIP methods provide a systematic approach to search for the gait pattern, where they use a tree-search algorithm to explore the motion plan with different gait patterns. However, MIP methods often feature expensive computation, which is not suitable for online computation. To address the computation burden induced by the gait pattern selection issue, in this thesis, we propose to build a gait pattern selection map through offline computations. By using this map, we can quickly decide the gait pattern during online planning.

3

PROBLEM DESCRIPTION

In this chapter, we provide a formal description of the multi-contact motion planning problem, and we also highlight the key computational challenges that can prevent online computation.

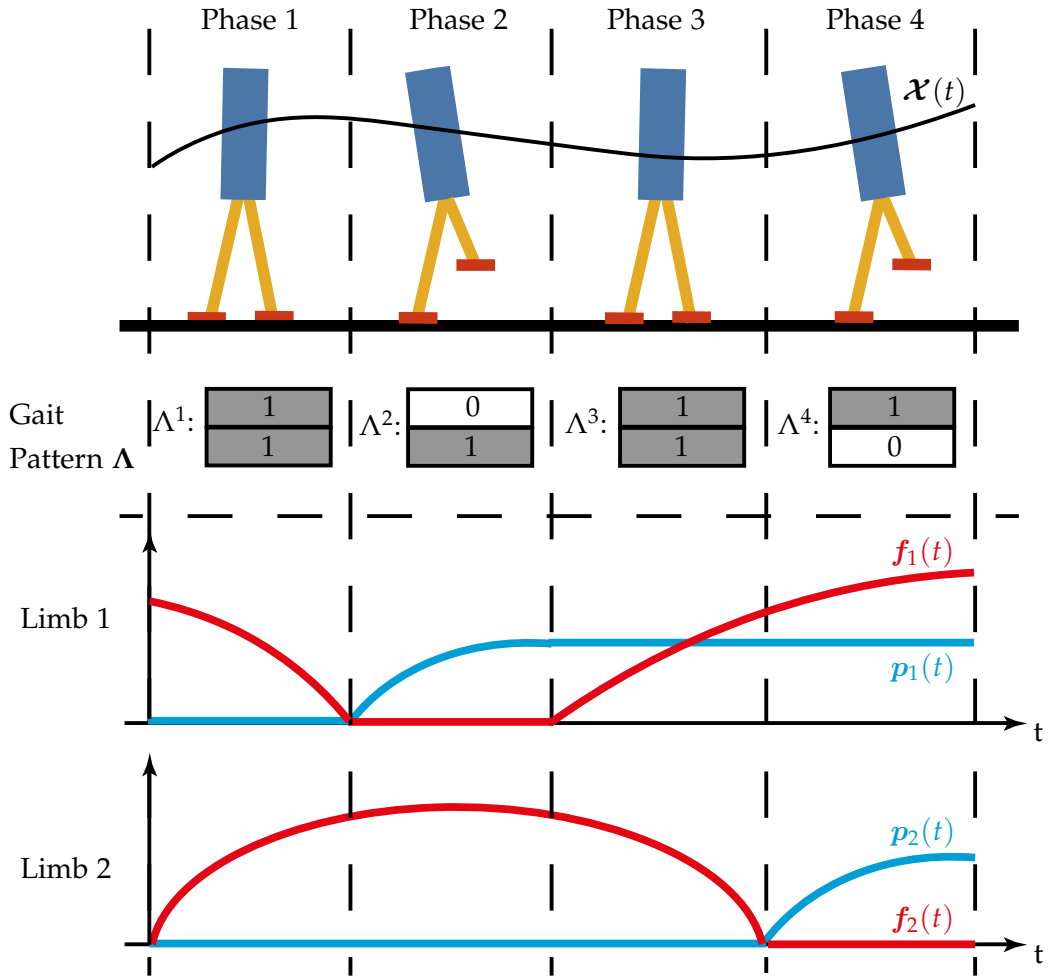


Figure 9: The multi-contact motion planning problem necessarily requires the resolution of both discrete and continuous decisions. The discrete decision is the gait pattern $\mathbf{\Lambda}$, which reflects the contact configuration Λ^q of each contact phase, e.g., we use 0 and 1 to indicate whether a foot is in active contact or not. The continuous decisions include the robot state trajectory $\mathcal{X}(t)$, the feet trajectories $\boldsymbol{p}_1(t)$, and $\boldsymbol{p}_2(t)$ and the control input trajectory (contact force profile) $\boldsymbol{f}_1(t)$ and $\boldsymbol{f}_2(t)$. Furthermore, to ensure the motion plan is dynamically feasible, we also need to consider feasibility constraints, i.e., the complex nonlinear dynamics constraint, the kinematics constraint, and the contact condition constraint characterized by the gait pattern (no force can be exerted if the foot is not in contact, the foot also cannot move if it is in active contact).

Without loss of generality, let us consider a legged system with N_L limbs. In general words, the multi-contact motion planning problem can be described as follows: given a task specification κ_{task} (for example, a goal state) and an environment model $\mathbf{\Omega}$, we aim to find a motion plan over a finite horizon $t \in [0, T]$, which tells how the legged robot can achieve the desired task by utilizing the contacts between its limbs and the environment.

As illustrated in Fig. 9, given the discrete and non-smooth nature of the contact events, we can divide the multi-contact motion plan into N_{ph} contact phases. In this thesis, we consider the number of contact phases N_{ph} as a hyper-parameter of the motion plan and it is pre-specified by the user. In each phase, the robot exhibits a contact configuration (which limbs are in contact with the ground and which limbs are not in contact) that is different from the ones of adjacent phases. These contact configurations can be seen as a structure of the motion plan, which also characterizes the feasibility constraints imposed in each phase. To obtain such a multi-phase motion plan, we typically need to resolve both discrete and continuous decisions.

To give more detail, the discrete decision refers to the issue of choosing a gait pattern that specifies the contact configurations of all contact phases. More specifically, in this thesis, we model the contact configuration of each phase $q \in \{1, ..., N_{ph}\}$ as $\Lambda^q = \{0, 1\}^{N_L}$, where we use 1 to represent a foot is in contact with the ground, and 0 to indicate a foot is detached from the ground. Then, we denote the gait pattern as $\mathbf{\Lambda} = [\Lambda^1, ..., \Lambda^{N_{ph}}]$, which collects the contact configuration of all contact phases. We illustrate an example gait pattern in Fig. 9. As we will discuss later, finding an appropriate gait pattern is important, as a change in the gait pattern also switches the feasibility constraints imposed on the multi-contact motion plan.

On the other hand, the continuous decision of the multi-contact motion planning problem is to synthesize the contact and motion plan that complies with the chosen gait pattern. To represent the contact and motion plan, we can use the following task-space quantities, which include:

- The robot state trajectory $\boldsymbol{\mathcal{X}}(t)$.
- The feet trajectory $\boldsymbol{\mathcal{P}}(t) = [\boldsymbol{p}_1(t), ..., \boldsymbol{p}_{N_L}(t)]^T$ that collects the foot trajectory $\boldsymbol{p}_l(t)$ of all feet $l \in \{1, ..., N_L\}$.
- The control trajectory $\boldsymbol{\mathcal{U}}(t) = [\boldsymbol{f}_1(t), ..., \boldsymbol{f}_{N_L}(t)]^T$ that includes the contact force profile $\boldsymbol{f}_l(t)$ of all feet $l \in \{0, ..., N_L\}$.

In the meantime, to ensure the motion plan is dynamically feasible, multi-contact motion planning necessarily requires the consideration of feasibility constraints. For example, the feet trajectory and the robot state trajectory should be consistent with

the kinematics constraint of the robot. Furthermore, the state evolution of the robot should also respect the system dynamics constraint. The feet trajectory and the control trajectory (contact forces) should also meet the contact condition, i.e., the foot can exert pushing force, only if it is staying in contact with the ground.

To model this multi-contact motion planning problem, we can formulate it as an optimal control problem, whose general form can be written as:

$$\min_{\boldsymbol{\mathcal{X}}(t),\boldsymbol{\mathcal{P}}(t),\boldsymbol{\mathcal{U}}(t),\boldsymbol{\Lambda}} \quad \int_0^T J(\boldsymbol{\mathcal{X}}(t),\boldsymbol{\mathcal{P}}(t),\boldsymbol{\mathcal{U}}(t),\boldsymbol{\Lambda})\, dt \tag{1a}$$

$$\text{s.t.} \quad \forall t \in [0,T]:$$

$$\boldsymbol{h}_{Task}(\boldsymbol{\mathcal{X}}(t),\boldsymbol{\mathcal{P}}(t),\boldsymbol{\mathcal{U}}(t),\boldsymbol{\kappa_{task}}) \leq 0, \quad \text{(Task)} \tag{1b}$$

$$\boldsymbol{h}_{Contacts}(\boldsymbol{\mathcal{P}}(t),\boldsymbol{\mathcal{U}}(t),\boldsymbol{\Omega} \mid \boldsymbol{\Lambda}) \leq 0, \quad \text{(Contacts)} \tag{1c}$$

$$\boldsymbol{h}_{Kinematics}(\boldsymbol{\mathcal{X}}(t),\boldsymbol{\mathcal{P}}(t)) \leq 0, \quad \text{(Kinematics)} \tag{1d}$$

$$\boldsymbol{h}_{Dynamics}(\boldsymbol{\mathcal{X}}(t),\boldsymbol{\mathcal{P}}(t),\boldsymbol{\mathcal{U}}(t),\boldsymbol{\Omega}) \leq 0, \quad \text{(Dynamics)} \tag{1e}$$

where $J(\cdot)$ is the user-defined cost function, (1b) is the task constraint that ensures the motion plan meets the desired task specification κ_{task}, (1c) is the contact condition constraint whose formulation is dependent on the chosen gait pattern $\boldsymbol{\Lambda}$, (1d) and (1e) are the robot kinematics and dynamics constraints.

Typically, solving the multi-contact planning problem (1) is a time-consuming process. The computation challenges mainly come from the following two aspects: i) combinatorial complexity that arises from the issue of selecting an appropriate gait pattern, and ii) high-dimensionality and non-convexity arise from the contact and motion planning. Next, we elaborate and discuss these two computational challenges in detail.

Challenge 1: Combinatorial Complexity due to Gait Pattern Selection

In multi-contact motion planning, choosing an appropriate gait pattern is important to the feasibility and optimality of the motion plan with respect to the given task. This is because the gait pattern specifies the contact configuration of each contact phase, which in turn determines the exact formulation of the contact condition constraint (1c) described as follows:

- If a foot is in contact with the ground, it can exert contact forces. However, at the same time, the velocity of the foot should be zero, and the foot should stay stationary (assuming sliding is not allowed).
- If a foot breaks contact with the ground, it can move in the free space, but the foot cannot exert any contact forces.

As we can imagine, different gait patterns can imply different instances of the contact condition constraint (1c) imposed on the feet trajectories and the contact force profiles, which can in turn affect the dynamics of the robot (1e). As a result, to find an appropriate gait pattern, we typically need to explore a wide range of motions characterized by different gait patterns. However, this brings combinatorial complexity into the planning problem.

Challenge 2: High-dimensionality and Non-convexity arises from the Contact and Motion Planning

In addition to the combinatorial complexity, another computational challenge of multi-contact motion planning is the high-dimensionality and non-convexity coming from the contact and motion planning, i.e., computing the robot state trajectory together with modulation of the feet trajectories and contact force profiles. To give more detail, when computing these trajectories, we necessarily need to consider the dynamics constraint (1e) to ensure the motion plan is dynamically feasible. However, as discussed in Section 2.1.2, existing dynamics models used for planning multi-contact motions are usually high-dimensional and nonlinear, i.e., the whole-body dynamics model features both high-dimensionality and nonlinearity, the centroidal dynamics model has reduced dimensionality but it is intrinsically nonlinear. As a result, planning multi-contact motions with these complex dynamics models often results in high-dimensional and non-convex problems that are computationally expensive.

Given these computational challenges, existing multi-contact motion planning methods often take a considerable amount of time to compute, and thus they are limited to offline usage. In this thesis, we focus on accelerating the computation speed of multi-contact motion planning. In particular, we are interested in achieving multi-contact motion planning in an online receding horizon fashion. To approach this goal, we start from the case where we compute the contact and motion plans with a pre-defined gait pattern. Although in this case, we avoid the combinatorial search of the gait pattern and the formulation of the contact condition (1c) is pre-determined, the resultant planning problem is still high-dimensional and non-convex

due to the consideration of complex nonlinear dynamics constraint (1e). To improve the computation efficiency, our strategy is to simplify the modeling of this planning problem. The detail of our approach is presented from Chapter 4 to Chapter 8. Then, in Chapter 9, we explore the possibility to address the gait pattern selection problem efficiently. To this end, we carry out offline computations to investigate how the task specification and environment model can influence the choice of the gait pattern. Throughout our exploration, we find that there exists a map from the task conditions to the gait pattern selection, and we can utilize this map to select the gait pattern during online planning.

4

BACKGROUND AND APPROACH OVERVIEW FOR ONLINE MULTI-CONTACT RECEDING HORIZON PLANNING

Starting from this chapter, we present the technical details of our approaches for achieving online Receding Horizon Planning (RHP) of contact and motion plans. We recall that, in this case, we focus on computing the robot motion with a pre-defined gait pattern.

In this chapter, we will focus on providing background for understanding our methods. More specifically, in Section 4.1, we give a general introduction to the problem of computing contact and motion plans in a receding horizon fashion, and we also briefly introduce the concept of our proposed methods for achieving online computation. The core idea of our methods is to accelerate the computation speed by finding computationally efficient approximations of the value function. Next, in Section 4.2, we provide a formal description of the RHP problem under the framework of Bellman's principle of optimality [7]. Furthermore, we also highlight the importance of considering a value function approximation for RHP, and introduce the principle of the traditional value function approximation approach, as well as our proposed methods for achieving more efficient value function approximation. Afterwards, we list the assumptions we made for our methods in Section 4.3. Lastly, we present the technical detail of the traditional RHP formulation (baseline) for computing contact and motion plans.

4.1 INTRODUCTION TO THE RHP PROBLEM AND OUR METHODS

Contact and motion planning usually requires the planner to compute a feasible CoM trajectory, along with a sequence of contact states (locations and timings). This problem is high-dimensional, nonlinear, and subject to discrete changes of dynamics that arise from breaking and making contacts [81, 91, 97, 109]. Traditionally, legged robot control methods often plan the contact and motion plans offline, and then track them with a controller [94, 109]. However, when deploying legged robots in the real world, robots can encounter environment changes and state drifts. These perturbations can cause the pre-planned motion to become invalid, and online motion (re)-planning is needed [77, 94].

Towards this end, RHP [68, 77] can be a promising solution. Similar to MPC [24, 28, 62, 73], Receding Horizon Planning (RHP) aims to constantly update the motion plan for immediate execution based on the state of the robot and the environment. This is often achieved by solving a finite-horizon TO problem [8], where the planning horizon consists of an execution horizon that plans the optimal actions for execution, and a prediction horizon that foresees the future (Fig. 10). Although RHP frameworks never execute the prediction horizon, it plays an important role to the success of RHP. Viewing the RHP problem from Bellman's perspective [7], the prediction horizon can be seen as an approximation of the value function—which predicts the feasibility and the future effort for achieving the task starting from a given robot state. Such a value function is responsible for guiding the execution horizon to plan optimal actions that are beneficial for the future operation of the robot.

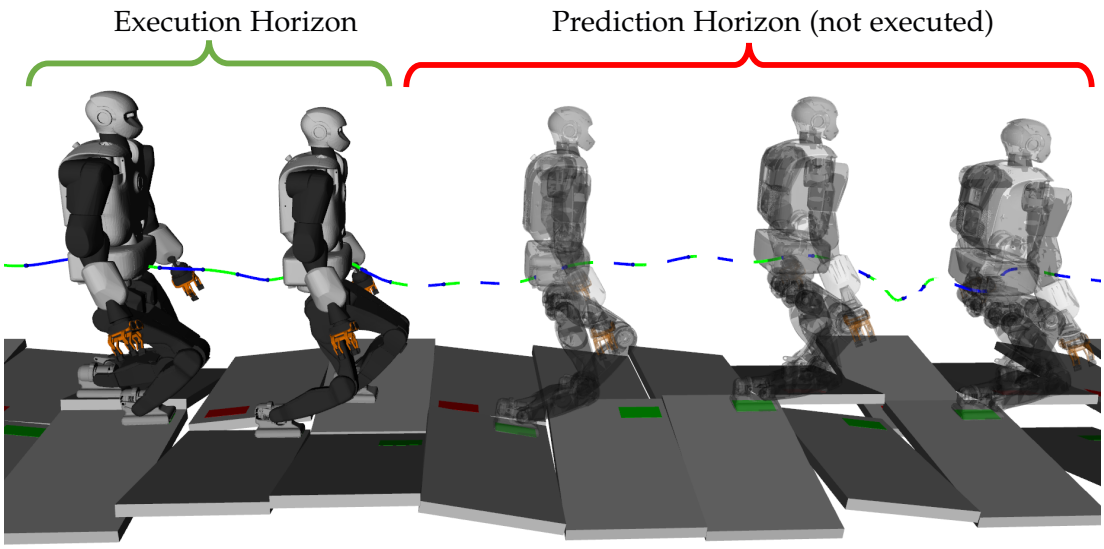


Figure 10: RHP frameworks usually compute the motion plan with an extended planning horizon. This includes an execution horizon that plans the motion for immediate execution, as well as a prediction horizon (not executed) that foresees the future beyond the execution horizon. The prediction horizon serves as an approximation of the value function, which guides the executing horizon by predicting whether the decisions made in the execution horizon can facilitate the completion of the task or not.

Traditionally, RHP frameworks often compute the trajectory over the entire planning horizon with an accurate dynamics model. This ensures the execution horizon is

always dynamically consistent, while in the meantime allowing the prediction horizon to approximate the value function as accurately as possible. However, planning the prediction horizon with an accurate model can result in expensive computation, especially when a long planning horizon and complex dynamics need to be considered, i.e., planning dynamic motions to traverse large obstacles.

To accelerate the computation speed, we explore the possibility to reduce the computation burden required for approximating the value function. Following this idea, we propose a trajectory-based approach and a learning-based approach that can achieve value function approximation with improved computational efficiency.

More specifically, our trajectory-based approach—which we call RHP with Multiple Levels of Model Fidelity—follows the traditional formalism that models the value function with the prediction horizon. However, instead of considering accurate dynamics models, we relax the model accuracy when planning the prediction horizon. This allows us to reduce the overall complexity of the RHP problem.

Alternatively, we can approximate the value function with a learned model [118]. However, learning a value function for the contact and motion planning problem can be challenging. The main difficulty is that the value function is defined in a coupled state-environment space, which requires a flexible representation to capture the landscape changes of the value function with respect to different environments [25]. In this thesis, we circumvent this issue by learning an oracle that can predict a local objective—an intermediate goal state towards the completion of a given task—based on the current robot state, goal position, and the environment model. We then construct a local value function to guide a short-horizon TO to plan the execution horizon towards the predicted local objectives. We refer to this approach as Locally-Guided Receding Horizon Planning (LG-RHP).

In the next section, we focus on providing a formal description of the RHP problem and highlighting the fact that considering an approximation of the value function is important to the success of RHP. In addition, we also formally describe the principle on how the value function is approximated in the traditional RHP as well as our proposed multi-fidelity RHP and locally-guided RHP.

4.2 VALUE FUNCTION APPROXIMATION FOR RECEDING HORIZON PLANNING

In this section, we describe the problem formulation of RHP based on a generic dynamical system and we illustrate the importance of incorporating an approximated value function based on the Bellman's principle of optimality [7]. Moreover, we also summarise the different techniques for approximating the value function. These include: 1) the traditional RHP approach which approximates the value function by

computing the prediction horizon with an accurate dynamics model; 2) our multi-fidelity RHP approach which models the value function with a prediction horizon while considering relaxed system dynamics models; 3) our locally-guided RHP that learns a value function model from past experiences.

Without loss of generality, let us denote by $\boldsymbol{x} \in \mathbb{R}^n$ the robot state and $\boldsymbol{u} \in \mathbb{R}^m$ the control input. In RHP, each cycle is required to compute a motion plan for immediate execution in the next cycle. We define such a motion plan is composed of a state trajectory $\{\boldsymbol{x}_0, ..., \boldsymbol{x}_T\}$ starting from a given initial state $\boldsymbol{x}_0$, and a control trajectory $\{\boldsymbol{u}_0, ..., \boldsymbol{u}_T\}$. To compute this motion plan, RHP frameworks usually need to solve a TO problem with the general form complying with the Bellman's equation [7]:

$$\min_{\substack{\boldsymbol{x}_0,...,\boldsymbol{x}_T,\\ \boldsymbol{u}_0,...,\boldsymbol{u}_T}} \quad \sum_{t=0}^{T-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t) + V(\boldsymbol{x}_T) \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \mathcal{F}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{2b}$$

where $l(\cdot)$ is the running cost on the state and control, $V(\cdot)$ is the value function evaluated at the terminal state of the state trajectory $\boldsymbol{x}_T$, and (2b) is the system dynamics constraint, and $\mathcal{F}(\cdot)$ represents the discrete-time dynamics of the robot. As Bellman suggests, the optimal policy—approximated by the motion plan to be executed—should not only minimize its running cost l, but also lead to a state $\boldsymbol{x}_T$ that optimizes the value function $V(\boldsymbol{x}_T)$.

By definition, the value function is modeled as the optimal cost of an infinite-horizon trajectory starting from $\boldsymbol{x}_T$ till the completion of the task, while respecting the system dynamics constraint (Fig. 11-a):

$$V(\boldsymbol{x}_T) = \min_{\substack{\boldsymbol{x}_T,...,\boldsymbol{x}_\infty,\\ \boldsymbol{u}_T,...,\boldsymbol{u}_\infty}} \quad \sum_{t=0}^{\infty} l(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{3a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \mathcal{F}(\boldsymbol{x}_t, \boldsymbol{u}_t). \tag{3b}$$

The value function $V(\boldsymbol{x}_T)$ reflects the feasibility and the future effort required for accomplishing the given task starting from any chosen state $\boldsymbol{x}_T$. Such a value function can provide gradients to direct the optimal policy (motion to be executed) towards a state $\boldsymbol{x}_T$ that is favorable for the future operation of the robot. However, evaluating the value function with an infinite-horizon trajectory is non-trivial, and hence we need approximations.

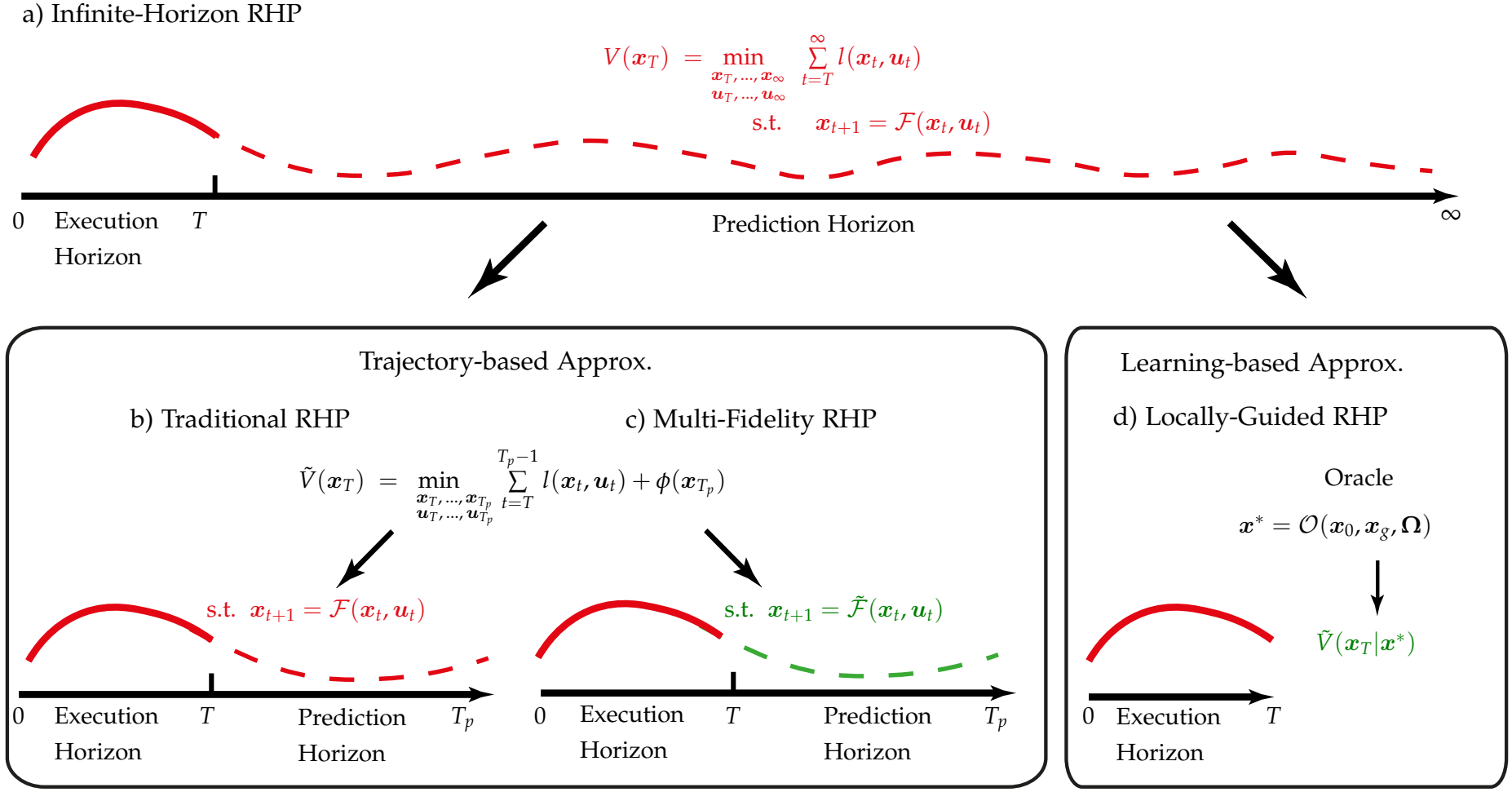


Figure 11: a) Infinite-horizon RHP problem that models the value function with the prediction horizon of an infinite length; b) Traditional RHP approach which approximates the value function by considering a finite-length prediction horizon (from time T to T_p). Nevertheless, traditional RHP struggles to compute online, as the prediction horizon considers an accurate dynamics model (usually non-convex); c) Multi-fidelity RHP, where we improve the computation efficiency by relaxing the model accuracy in the prediction horizon; d) Locally-Guided RHP shortens the planning horizon by approximating the value function with a learned model.

Traditionally, RHP frameworks approximate the value function by considering a finite-horizon trajectory starting from the time T to $T_p \ll \infty$:

$$\tilde{V}(\boldsymbol{x}_T) = \min_{\substack{\boldsymbol{x}_T,\dots,\boldsymbol{x}_{T_p},\\ \boldsymbol{u}_T,\dots,\boldsymbol{u}_{T_p}}} \sum_{t=T}^{T_p-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t) + \phi(\boldsymbol{x}_{T_p}), \tag{4a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \mathcal{F}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{4b}$$

where the optimal cost from the time T_p to infinity is lumped into the terminal cost term $\phi(\boldsymbol{x}_{T_p})$. Typically, the terminal cost function is defined as a quadratic term that attracts the terminal state $\boldsymbol{x}_{T_p}$ towards a user-defined goal state $\boldsymbol{x}_g$:

$$\tilde{V}(\boldsymbol{x}_{T_p}) = (\boldsymbol{x}_{T_p} - \boldsymbol{x}_g)^\top (\boldsymbol{x}_{T_p} - \boldsymbol{x}_g). \tag{5}$$

By combining (4a) into (2a), we can achieve a TO problem with an extended planning horizon (Fig. 11-b):

$$\min_{\substack{\boldsymbol{x}_0,\dots,\boldsymbol{x}_{T_p},\\ \boldsymbol{u}_0,\dots,\boldsymbol{u}_{T_p}}} \underbrace{\sum_{t=0}^{T-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t)}_{\text{Optimal Policy (EH)}} + \underbrace{\sum_{t=T}^{T_p-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t) + \phi(\boldsymbol{x}_{T_p})}_{\text{Value Function Approximation (PH)}} \tag{6a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \mathcal{F}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{6b}$$

where we split the planning horizon into an Execution Horizon (EH) that computes optimal policy (the motion plan to be executed) from the time 0 to T, and a Prediction Horizon (PH) that approximates the value function by computing trajectories from the time T to T_p.

Although (6a) has a finite planning horizon, online computation is still challenging for complex dynamical systems such as legged robots. The computation complexity mainly comes from the planning of the prediction horizon under the consideration of the nonlinear dynamics constraints (8b), which increases the dimensionality and non-convexity of an already challenging problem. In this part of the thesis, the traditional RHP approach is considered as the baseline.

To improve the computation efficiency, a promising direction is to mitigate the computation burden required for value function approximation. In this thesis, we present two novel methods that can approximate the value function with reduced computation complexity.

Our first method follows the trajectory-based formalism that approximates the value function by computing a prediction horizon that looks into the future. However, instead of considering an accurate system dynamics constraint (usually non-convex) in the prediction horizon, we propose to plan the prediction horizon with a relaxed system dynamics model. This gives rise to a novel type of TO formulation that features a planning horizon with multiple levels of model fidelity (Fig. 11-c):

$$\min_{\substack{\boldsymbol{x}_0,\dots,\boldsymbol{x}_{T_p},\\ \boldsymbol{u}_0,\dots,\boldsymbol{u}_{T_p}}} \underbrace{\sum_{t=0}^{T-1} l(\boldsymbol{x}_t,\boldsymbol{u}_t)}_{\text{EH}} + \underbrace{\sum_{t=T}^{T_p-1} l(\boldsymbol{x}_t,\boldsymbol{u}_t) + \phi(\boldsymbol{x}_{T_p})}_{\text{PH}}, \tag{7a}$$

$$\text{s.t.} \quad \forall t \in [0,T]:$$

$$\boldsymbol{x}_{t+1} = \mathcal{F}(\boldsymbol{x}_t,\boldsymbol{u}_t), \quad \text{(Accurate Model)} \tag{7b}$$

$$\forall t \in [T,T_p]:$$

$$\boldsymbol{x}_{t+1} = \tilde{\mathcal{F}}(\boldsymbol{x}_t,\boldsymbol{u}_t), \quad \text{(Relaxed Model)} \tag{7c}$$

where the execution horizon remains to compute the optimal polity with the accurate dynamics model $\mathcal{F}$ (7b), while the prediction horizon approximates the value function with the relaxed dynamics model $\tilde{\mathcal{F}}$ (7c). We call this approach as Receding Horizon Planning with Multiple Levels of Model Fidelity or Multi-Fidelity Receding Horizon Planning (MF-RHP) for short. Compared to the traditional TO formalism (6a), our multi-fidelity RHP ensures the execution horizon (to be executed) is always dynamically consistent and thus can be tracked by a controller. In the meantime, our multi-fidelity RHP also features reduced overall computation complexity, owing to the relaxed model accuracy in the prediction horizon. In this thesis, we present and test three candidate multi-fidelity RHPs, where the prediction horizon considers different convex relaxations of the centroidal dynamics model.

Alternatively, another option for approximating the value function is to learn a parametric model $\tilde{V}(\boldsymbol{x}|\boldsymbol{\theta})$ from the past experience [118]. Given this learned value function model, we can shorten the planning horizon to only cover the execution horizon:

$$\min_{\substack{\boldsymbol{x}_0,\dots,\boldsymbol{x}_T,\\ \boldsymbol{u}_0,\dots,\boldsymbol{u}_T}} \underbrace{\sum_{t=0}^{T-1} l(\boldsymbol{x}_t,\boldsymbol{u}_t)}_{\text{Optimal Policy (EH)}} + \underbrace{\tilde{V}(\boldsymbol{x}_T|\theta)}_{\text{Learned Value Function}}, \tag{8a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \mathcal{F}(\boldsymbol{x}_t,\boldsymbol{u}_t), \tag{8b}$$

However, learning a value function model for the contact and motion planning problem can be challenging. The difficulty mainly comes from the consideration of the environment model. This introduces the challenge of finding a flexible parameterization that can represent the value function in the coupled state-environment space [25]. To tackle this issue, we propose to learn an oracle $\mathcal{O}$ that can predict intermediate goal states $\boldsymbol{x}^*$ for completing a given task, based on current robot state $\boldsymbol{x}_0$, the final goal state $\boldsymbol{x}_g$, and the environment model $\boldsymbol{\Omega}$:

$$\boldsymbol{x}^* = \mathcal{O}(\boldsymbol{x}_0, \boldsymbol{x}_g, \boldsymbol{\Omega}). \tag{9}$$

We refer to these intermediate goal states as local objectives, and we use them to construct local value functions:

$$\tilde{V}(\boldsymbol{x}_T|\boldsymbol{x}^*) = (\boldsymbol{x}_T - \boldsymbol{x}^*)^\top (\boldsymbol{x}_T - \boldsymbol{x}^*). \tag{10}$$

Then, we use these local value functions to guide the short-horizon TO (8a) to plan the execution horizon towards the predicted local objectives $\boldsymbol{x}^*$. We call this approach as Locally-Guided Receding Horizon Planning (LG-RHP) and illustrate the idea in (Fig. 11-d).

In this section, we illustrate the importance of considering a value function approximation in RHP from Bellman's perspective. Furthermore, we also present the principles of different value function approximation methods employed in traditional RHP as well as our proposals—multi-fidelity RHP and locally-guided RHP. However, all the descriptions are based on a generic dynamics model. In the rest of the thesis, we focus on providing technical detail on how to apply these RHP approaches in the context of planning contact and motion plans.

4.3 ASSUMPTIONS

In this section, we describe the assumptions we made for our methods:

1. We focus on computing the centroidal motion plan of a humanoid robot walking on uneven terrain. The decision variables include a centroidal trajectory of the CoM, along with a sequence of contact locations and timings.

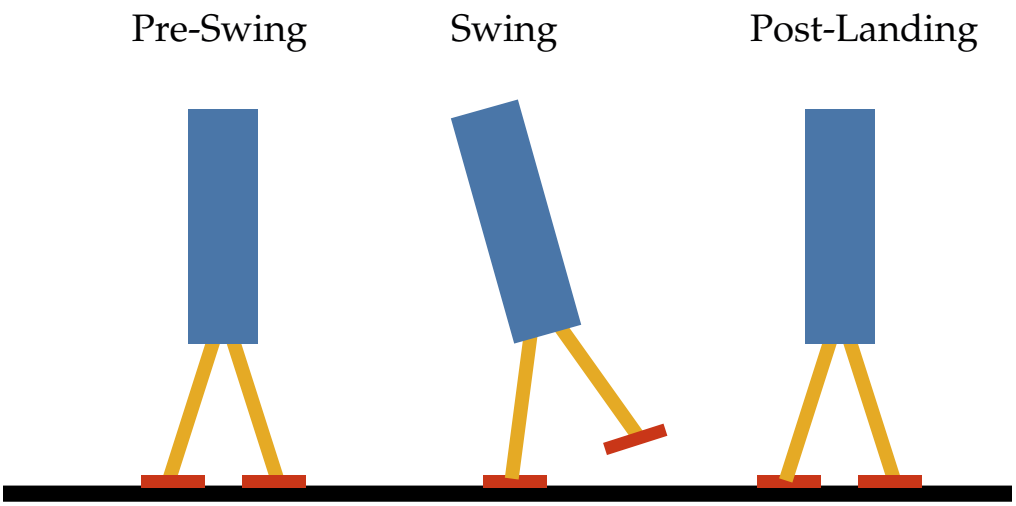


Figure 12: We define each step is composed of three phases: pre-swing (double support), swing (single support), and post-landing (double support)

2. We define each step consists of three phases (Fig. 12): i) pre-swing (double support), ii) swing (single support), and iii) post-landing (double support). This leads to a multi-phase TO formulation, which features phase-specific constraints (dynamics and kinematics) determined by the contact configuration of each phase.

3. We define that the execution horizon always covers the motion plan for making a single step (the first three phases), while the prediction horizon can plan ahead for multiple steps.

4. We neglect the impact dynamics (state jumps when creating contacts) in our formulation. This is valid for the walking gait, as the swing foot can make contact with near-zero velocity.

5. We model the robot feet as rectangular patches. As commonly done, we model each vertex of the rectangle as a contact point.

6. We approximate the kinematics constraints of the CoM and the relative positions of the contacts as convex polytopes (see an example in Fig. 13). To generate these polytopes, we take the approach described in [95]. More specifically, we firstly offline sample a large number of robot configurations, from which we can extract CoM positions and foot locations in a given end-effector frame. Then,

we compute the polytopes as the convex hull of these CoM positions and foot locations.

7. We model the environment as a set of rectangular contact surfaces. We pre-define the sequence of these contact surfaces in which the swing foot will land upon, while the contact locations (within the surfaces) and the contact timings are optimized.

8. The swing foot trajectory is interpolated after we compute the centroidal motion plan. This is achieved by connecting the planned contact locations with a spline. Although the kinematics constraint is not imposed when generating the swing foot trajectory, we enforce the swing foot to move with a similar speed of the planned CoM trajectory. This can increase the chance of obtaining a swing foot trajectory that is kinematically feasible, and we find that the swing foot trajectory generated with this approach is always kinematically feasible during our experiment.

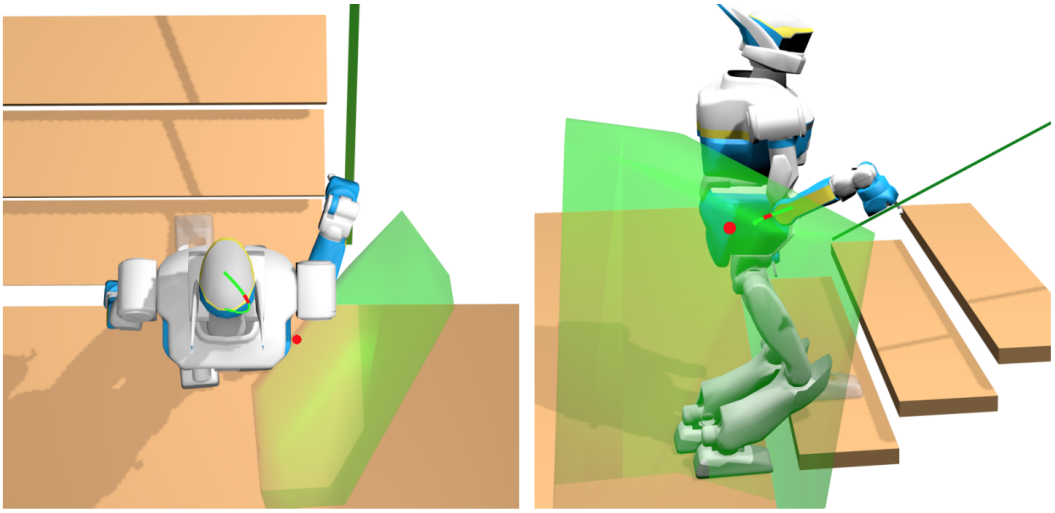


Figure 13: An example of the approximated kinematics constraints. The CoM is constrained in a convex reachability polytope (green) with respect to the stance foot. The same idea also applies to the relative footstep reachability, where each footstep needs to stay inside the rechability polytope defined by the previouse footstep. The figure takes HRP-2 as an example, and we apply the same approach for Talos. The picture is adapted from [35].

4.4 TRADITIONAL RECEDING HORIZON PLANNING (RHP) APPROACH FOR COMPUTING CONTACT AND MOTION PLANS

In this section, we introduce the traditional RHP approach for computing contact and motion plans for a humanoid robot walking on uneven terrain, and we consider the centroidal dynamics model to govern the motion. Typically, traditional RHP computes the motion plan by solving a TO problem while considering an accurate dynamic model over the entire horizon. This traditional RHP is considered as the baseline of our work. Next, we present the definition of the decision variables of the TO problem in Section 4.4.1 as well as the complete TO formulation in Section 4.4.2. Lastly, we discuss the computation challenges of the traditional RHP approach in Section 4.4.3.

4.4.1 *Decision Variables*

In each planning cycle, given a finite planning horizon of n steps, an initial robot state $\boldsymbol{x}_{init}$, a final goal state $\boldsymbol{x}_g$, and a sequence of contact surfaces $\{\mathcal{S}_1, ..., \mathcal{S}_n\}$ that the robot will step upon, the RHP framework aims to compute a multi-phase motion plan consists of a state trajectory $\boldsymbol{\mathcal{X}}$, a control trajectory $\boldsymbol{\mathcal{U}}$, a sequence of contact locations $\boldsymbol{\mathcal{P}}$ and a list of phase switching timings $\boldsymbol{\mathcal{T}}$. We elaborate the definition of these decision variables as follows:

- $\boldsymbol{\mathcal{X}} = \{X^1, ..., X^{N_{ph}}\}$
  State trajectory X^q of all phases $q \in \{1, ..., N_{ph}\}$. In each phase, we discretize the state trajectory into N_k knots: $X^q = \{\boldsymbol{x}_1^q, ..., \boldsymbol{x}_{N_k}^q\}$. We denote the state vector as $\boldsymbol{x} = [\boldsymbol{c}^\top, \dot{\boldsymbol{c}}^\top, \boldsymbol{L}^\top]^\top$, where $\boldsymbol{c} \in \mathbb{R}^3$ is the CoM position, $\dot{\boldsymbol{c}} \in \mathbb{R}^3$ is the CoM velocity, $\boldsymbol{L} \in \mathbb{R}^3$ is the total angular momentum expressed at the CoM.

- $\boldsymbol{\mathcal{U}} = \{U^1, ..., U^{N_{ph}}\}$
  Control input trajectory of all phases. Same as the state trajectory, we discretize each phase of the control trajectory into N_k knots: $U^q = \{\boldsymbol{u}_1^q, ..., \boldsymbol{u}_{N_k}^q\}$. The control input vector is defined as $\boldsymbol{u} = [\boldsymbol{f}_1^\top, ..., \boldsymbol{f}_{N_c}^\top]^\top$ which collects the contact force $\boldsymbol{f}_c \in \mathbb{R}^3$ of all contact points $c \in \{1, ..., N_c\}$.

- $\boldsymbol{\mathcal{P}} = \{\boldsymbol{p}^1, ..., \boldsymbol{p}^n\}$
  A sequence of footstep locations (center of the foot), where $\boldsymbol{p}^i \in \mathbb{R}^3$ denotes the contact location of the i-th step. The orientation of each footstep is defined as a constant, where the roll and the pitch are given by the orientation of the corresponding contact surface $\mathcal{S}_i$, and the yaw is set to zero degrees.

- $\mathcal{T} = \{t^1, ..., t^{N_{ph}}\}$
  A list of phase switching timings, where t^q indicates the timing when the motion plan switches from phase q to phase $q+1$. Based on these phase switching timings, we can define the time step of each phase q as $\tau^q = (t^q - t^{q-1})/N_k$.

4.4.2 *Traditional Trajectory Optimization (TO) Formulation for Computing Contact and Motion Plans*

To compute the motion plan, the traditional RHP usually solves a TO problem given by:

$$\min_{\mathcal{X},\mathcal{U},\mathcal{T},\mathcal{P}} \quad \sum_{q=1}^{N_{ph}} J^q(X^q, U^q) + \phi(\boldsymbol{x}_T) \tag{11a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{init} \tag{11b}$$

$$0 \leq t^1 \leq \cdots \leq t^{N_{ph}} \leq T_{max} \tag{11c}$$

$$\forall i \in \{1, ..., n\}:$$

$$\boldsymbol{p}^i \in \mathcal{S}_i \tag{11d}$$

$$\boldsymbol{p}^i \in \mathcal{R}_i \tag{11e}$$

$$\forall q \in \{1, ..., N_{ph}\}, \forall k \in \{1, ..., N_k\}:$$

$$\boldsymbol{c}_k^q \in \mathcal{K}_l^q, \quad \forall l \in \mathcal{L}_{cnt}^q \tag{11f}$$

$$\boldsymbol{x}_{k+1}^q = \mathcal{F}^q(\boldsymbol{x}_k^q, \boldsymbol{u}_k^q). \tag{11g}$$

Cost Terms

The cost function of (11a) includes the running cost J^q of each phase and the terminal cost $\phi(\boldsymbol{x}_T)$. We define the running cost:

$$J^q = \sum_{k=1}^{N_k} \tau^q (\ddot{\boldsymbol{c}}_k^{q\top} \ddot{\boldsymbol{c}}_k^q + \boldsymbol{L}_k^{q\top} \boldsymbol{L}_k^q), \tag{12}$$

which encourages the TO to generate smooth trajectories by penalizing large CoM accelerations and angular momentum. The terminal cost is defined as:

$$\phi(\boldsymbol{x}_T) = (\boldsymbol{x}_T - \boldsymbol{x}_g)^\top (\boldsymbol{x}_T - \boldsymbol{x}_g), \tag{13}$$

which attracts the terminal state $\boldsymbol{x}_T$ to approach the final goal state $\boldsymbol{x}_g$.

Constraints

To ensure the motion is dynamically consistent, we introduce constraints (11b)-(11g) described as follows:

1. Initial condition constraint (11b) enforces the state trajectory to start from the given initial state $\boldsymbol{x}_{init}$.

2. Phase switching timing constraint (11c) guarantees the phase switching timings t^q to increase monotonically and bounds the maximum motion duration $t^{N_{ph}}$ by T_{max}.

3. On-surface constraint (11d) restricts each contact location $\boldsymbol{p}^i$ to stay on the pre-assigned contact surface:

$$\mathcal{S}_i = \{\boldsymbol{p} \in \mathbb{R}^3, \boldsymbol{d}_i^T \boldsymbol{p} = 0, S_i \boldsymbol{p} \leq s_i\}, \tag{14}$$

   where the equality sets the plane containing the surface, and the inequalities define the boundaries of the surface.

4. Relative footstep constraint (11e) implements the relative reachability constraint of footsteps, where each contact location $\boldsymbol{p}^i$ is limited by a reachable workspace $\mathcal{R}_i$ with respect to the 6-D pose of the previous footstep $\boldsymbol{p}^{i-1}$. We represent the reachable workspace as a convex polytope:

$$\mathcal{R}_i : \{\boldsymbol{p}^i \in \mathbb{R}^3, \boldsymbol{R}_i(\boldsymbol{p}^i - \boldsymbol{p}^{i-1}) \leq \boldsymbol{r}_i\}, \tag{15}$$

   along with the orientation aligns to the posture of the previous footstep [87].

5. In each phase q, the CoM reachability constraint (11f) restricts the CoM position at k-th knot to stay within the reachable space $\mathcal{K}_l^q$ established by each foot l in active contact. Similarly, we approximate the reachable space as a convex polytope:

$$\mathcal{K}_l^q : \{\boldsymbol{c}_k^q \in \mathbb{R}^3, \boldsymbol{K}_l^q(\boldsymbol{c} - \boldsymbol{p}_l^q) \leq \boldsymbol{k}_l^q\}, \tag{16}$$

where $\boldsymbol{p}_l^q \in \mathbb{R}^3$ is the location of the active contact l in phase q. The orientations of these polytopes are also aligned to the pose of the active contacts [87].

6. Dynamics constraint (11g) imposes the system dynamics constraint on the trajectories. We approximate the integrals by the forward Euler integration scheme, and we consider the centroidal dynamics model [35, 75]:

$$\boldsymbol{c}_{k+1}^q = \boldsymbol{c}_k^q + \tau^q \dot{\boldsymbol{c}}_k^q, \tag{17a}$$

$$\dot{\boldsymbol{c}}_{k+1}^q = \dot{\boldsymbol{c}}_k^q + \tau^q \left(\frac{1}{m} \sum_{c \in \mathcal{C}^q} \boldsymbol{f}_{c,k}^q - \boldsymbol{g} \right), \tag{17b}$$

$$\boldsymbol{L}_{k+1}^q = \boldsymbol{L}_k^q + \tau^q \sum_{c \in \mathcal{C}^q} (\boldsymbol{p}_c - \boldsymbol{c}_k^q) \times \boldsymbol{f}_{c,k}^q, \tag{17c}$$

where m is the total mass of the robot, $\boldsymbol{g}$ is the gravitational acceleration, $\boldsymbol{p}_c$ is the location of each active contact point $c \in \mathcal{C}^q$ (the vertices of the rectangular foot). Furthermore, the contact forces are constrained by the linearized friction cone:

$$-\mu \boldsymbol{f}_c^{\hat{n}} \leq \boldsymbol{f}_c^{\hat{t}_1, \hat{t}_2} \leq \mu \boldsymbol{f}_c^{\hat{n}}, \tag{18}$$

where μ is the friction coefficient, $\boldsymbol{f}_c^{\hat{n}}$ and $\boldsymbol{f}_c^{\hat{t}_1, \hat{t}_2}$ are the normal and tangential components of the contact force, respectively.

4.4.3 *Discussion*

As mentioned in Section 4.2, to achieve successful RHP of contact and motion plans, transitional RHP necessarily requires the consideration of an extended planning horizon. That is, in addition to the execution horizon (the first step), the planning horizon also needs to incorporate a prediction horizon that foresees the future, i.e., the horizon starts from the second step till the last step. Although the prediction horizon is never executed, it serves as a trajectory-based approximation of the value function that can provide guidance to the execution horizon. However, traditional RHP typically plans the entire horizon with a single dynamics model. Given the

nonlinear centroidal dynamics constraint (17), considering an extended planning horizon can significantly increase the dimensionality and non-convexity of the TO, which hinders online computation. Moreover, the optimization of the contact timing (modulated by phase switching timings) also further increases the non-convexity of the TO problem. To achieve online multi-contact RHP, in the following chapters, we present the technical details of our methods to improve the computation efficiency of the TO, where we focus on exploring approximated value function models that can have reduced computation complexity.

4.5 CONCLUSION

In this chapter, we provided the background for understanding our approach to achieve online RHP of contact and motion plans. More specifically, we firstly gave a formal description of the RHP problem under the framework of Bellman's principle of optimality. In this formal description, we highlighted the importance of considering a value function approximation in RHP, and also introduced the concept of the traditional value function approximation approach, as well as our methods for achieving more efficient value function approximation. Afterwards, we listed the assumptions we made for our methods and presented the detail formulation of the traditional RHP in the context of computing contact and motion plans. This traditional RHP often struggles to achieve online computation, and it is considered as the baseline of our work. In the following chapters, we will focus on describing the technical details of our novel methods in achieving online RHP of contact and motion plans. The core idea of our methods is to explore approximated value function models that feature reduced computation burden.

5

MULTI-CONTACT RECEDING HORIZON PLANNING WITH MULTIPLE LEVELS OF MODEL FIDELITY

In this chapter, we present the technical details of our first approach for achieving online Receding Horizon Planning (RHP) of contact and motion plans. We call this approach as Receding Horizon Planning with Multiple Levels of Model Fidelity or Multi-Fidelity RHP for short. Compared to the traditional RHP approach, our multi-fidelity RHP reduces the computation burden by computing the prediction horizon with a convex relaxed model. We implement this idea in the context of planning dynamically consistent centroidal trajectories of humanoids walking on uneven terrain.

This chapter is organized as follows. In Section 5.1, we introduce the core idea of our multi-fidelity RHP approach and illustrate the main differences with respect to the traditional RHP. Afterwards, in Section 5.2, we present the three candidate convex relaxed models employed in the prediction horizon of our multi-fidelity RHP, and we compare their model complexity with respect to the centroidal model. Lastly, we conclude this chapter with a discussion presented in Section 5.3.

5.1 RATIONALE

In this section, we focus on introducing the important changes we made in multi-fidelity RHP with respect to the traditional RHP approach. These changes help us to reduce the overall computation complexity of the TO problem and facilitate online computation.

We recall that the traditional RHP (Fig. 14-a) usually employs a single-fidelity model over the entire look-ahead horizon (both the execution horizon and the prediction horizon). However, when computing contact and motion plans, we usually need to consider non-convex dynamics models, such as the centroidal dynamics model considered in our case. The centroidal dynamics model is intrinsically non-convex, as the cross products involved in the angular dynamics (17c) introduce bi-linear terms into the TO formulation. Moreover, as [80] indicates, the selection of the contact timing can also play an important role in achieving feasible motions. However, optimizing the contact timings (in our case, modulated by the phase switching timings) further increases the non-convexity of the traditional TO formulation. This is because the

variable phase switching timings result in varying time steps when integrating the system dynamics, which further increases the number of non-convex terms.

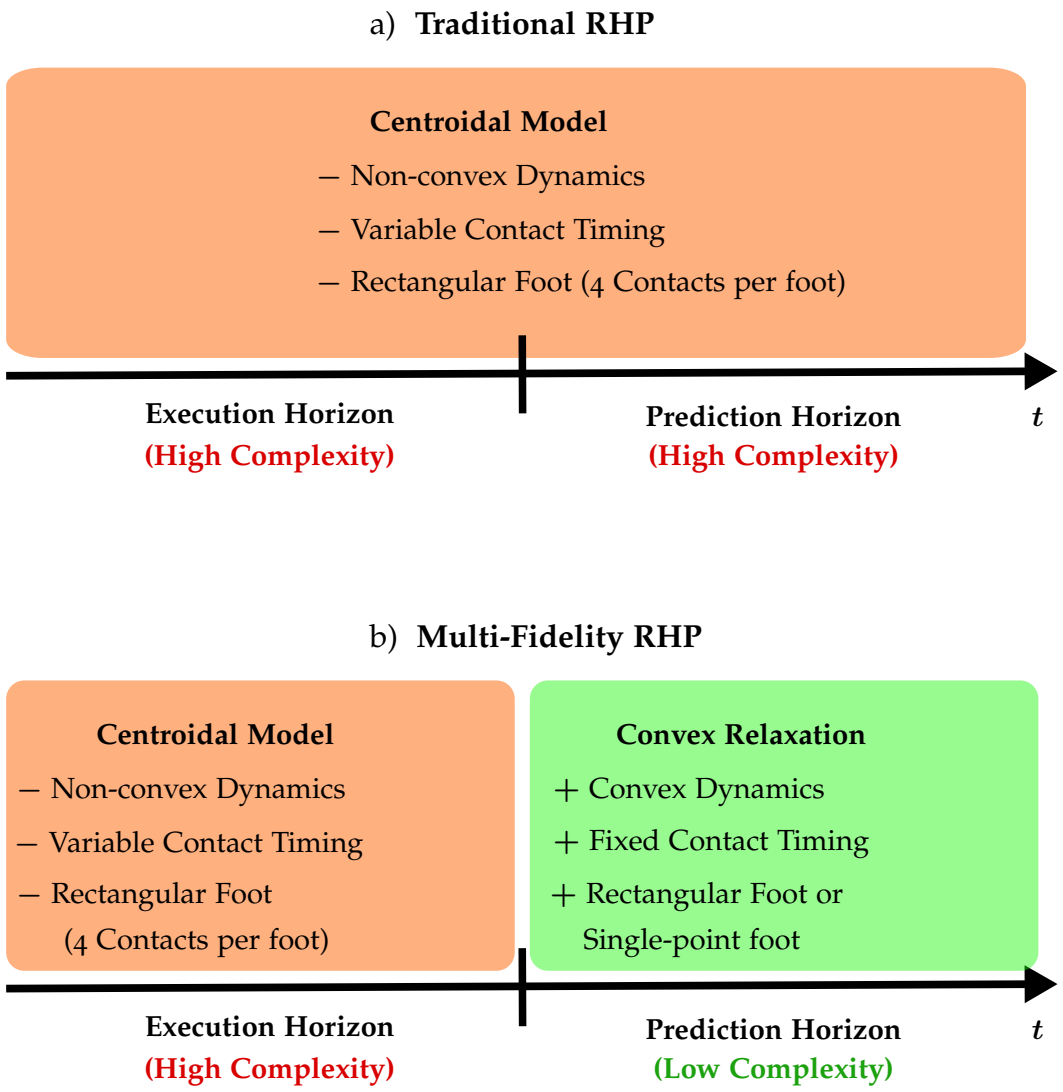


Figure 14: Complexity comparison between traditional RHP and our multi-fidelity RHP. We use orange to denote higher computation complexity, while green means lower computation complexity. Our multi-fidelity RHP formulation has reduced complexity due to the introduction of convex relaxations in the prediction horizon.

Considering the prediction horizon is never executed in RHP frameworks, while it serves as an approximation of the value function that is used to guide to planning of the execution horizon, we propose to improve the computation efficiency by simplifying

the model complexity in the prediction horizon. This leads to multi-fidelity RHP (Fig. 14-b) where we employ an accurate model of the system (in our case, the centroidal model) in the execution horizon, while having a convex relaxation in the prediction horizon. This setup ensures that in each RHP cycle, the motion for execution is dynamically consistent and can be tracked by a whole-body inverse dynamics controller, while in the meantime the overall problem complexity is reduced. The resultant TO problems are partially non-convex, and can be efficiently solved by off-the-shelf Non-Linear Programming (NLP) solvers.

As Fig. 14-b illustrates, to obtain a convex relaxed dynamics constraint in the prediction horizon, we introduce the following approximations:

1. We approximate the centroidal dynamics model with convex relaxations. In this thesis, we explore and test three convex relaxations of the centroidal dynamics model as described in Section 5.2.

2. To address the non-convexity that arises from the contact timing optimization, we fix the phase switching timings with empirical values in the prediction horizon (starting from the 4-th phase to the last phase).

Furthermore, we also explore to reduce the model dimensionality in the prediction horizon by switching the rectangular foot to a single-point foot.

Next, we focus on presenting the details of the three candidate convex relaxed models employed in the prediction horizon.

5.2 CANDIDATE CONVEX RELAXED MODELS FOR THE PREDICTION HORIZON

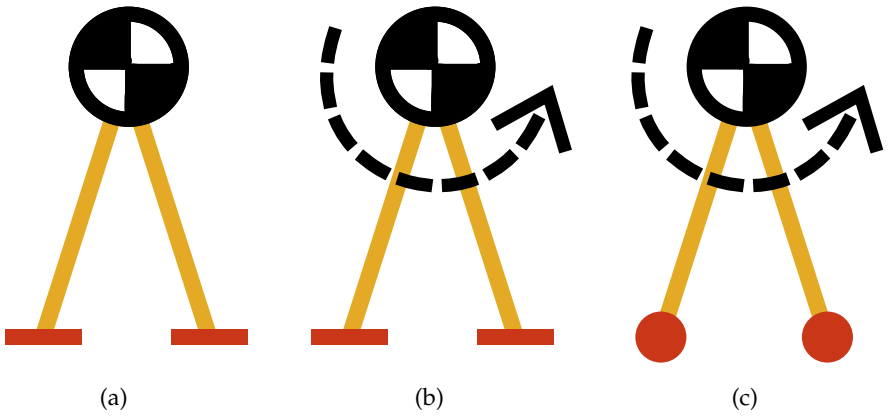
In this section, we introduce the modeling of the three convex relaxed models that are employed in the prediction horizon of multi-fidelity RHP. To provide an intuition of these candidate models, we illustrate their schematics in Fig. 15.

5.2.1 *Candidate 1: Linear CoM Dynamics*

In our first candidate (Fig. 15-a), the prediction horizon only considers the linear CoM dynamics defined by (17a)–(17b). This allows us to remove the non-convexity introduced by the angular dynamics (17c). As a result, in the prediction horizon, the state vector reduces to $\boldsymbol{x} = [\boldsymbol{c}^\top, \dot{\boldsymbol{c}}^\top]^\top$ and the running cost becomes:

$$J^q = \sum_{k=1}^{N_k} \tau^q (\ddot{\mathbf{c}}_k^{q\top} \ddot{\mathbf{c}}_k^q), \tag{19}$$

which only penalizes the CoM accelerations. As mentioned in Section 5.1, due to the contact timing optimization (modulated by the phase switching timings t^q), the linear CoM dynamics is still non-convex. To eliminate this non-convexity, we fix the phase switching timings $t^q, \forall q \in [4, N_{ph}]$ in the prediction horizon with empirical values. Furthermore, we remain to model the robot foot as a rectangular patch where each vertex is considered as a contact point.



(a) (b) (c)

Figure 15: Schematics of models used in the Prediction Horizon (PH): a) Candidate 1 only considers the linear CoM dynamics; b) Candidate 2 considers linear CoM dynamics as well as a convex relaxation of the angular dynamics of the centroidal model (dashed arrow), while modeling the robot foot as a rectangular patch; c) Candidate 3 employs the same modeling of the dynamics with respect to the Candidate 2, while switches the rectangular foot to a point foot.

5.2.2 *Candidate 2: Convex Angular Dynamics with Rectangular Contacts*

As we will present in Chapter 7, our first candidate usually fails to achieve successful RHP (fail to converge after a few cycles). This observation suggests that considering angular dynamics in the prediction horizon is important. As a result, in our second candidate, apart from having the linear CoM dynamics, we also include a convex outer approximation of the angular dynamics (17c) in the prediction horizon. This convex

outer approximation is based on the method described in [80]. Next, we introduce the detailed formulation of this convex outer approximation.

In the angular dynamics (17c) of the centroidal dynamics model, the non-convexity mainly comes from the bi-linear terms resulting from the expansion of the cross product between the lever arm $(\boldsymbol{p}_c - \boldsymbol{c})$ and the contact force vector $\boldsymbol{f}_c$. According to the principle described in [80], these bi-linear terms $\alpha\beta$ can be reformulated as the different of two quadratic terms $\psi^+ \in \mathbb{R}$ and $\psi^- \in \mathbb{R}$ (see an example in Fig. 16):

$$\alpha\beta = \psi^+ - \psi^-, \tag{20a}$$
$$\psi^+ = \frac{1}{4}(\alpha + \beta)^2, \tag{20b}$$
$$\psi^- = \frac{1}{4}(\alpha - \beta)^2. \tag{20c}$$

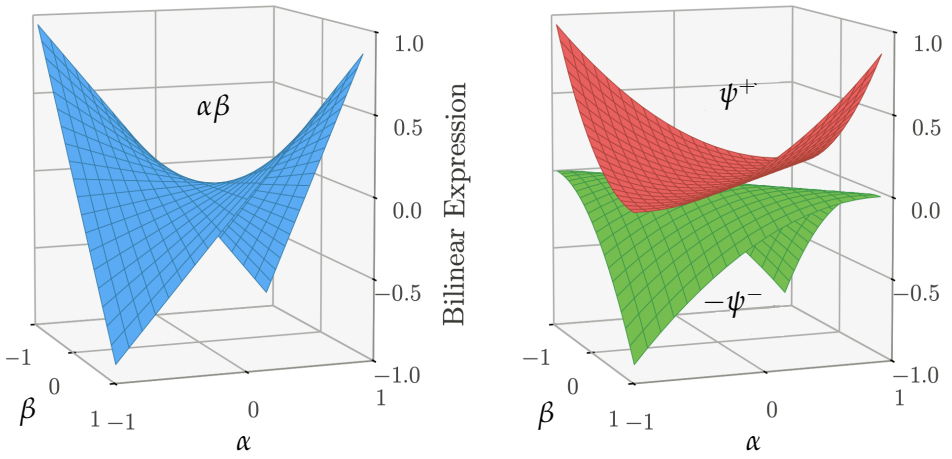


Figure 16: A bi-linear term $\alpha\beta$ can be decomposed into the difference of two quadratic terms $\psi^+ = \frac{1}{4}(\alpha + \beta)^2$ and $\psi^- = \frac{1}{4}(\alpha - \beta)^2$. The picture is modified from [80].

Nevertheless, replacing the bi-linear term $\alpha\beta$ with the quadratic terms ψ^+ and ψ^- will still result in a non-convex models. This is because (20b) and (20c) are quadratic equality constraints which are non-convex. To simplify the model, [80] suggests to approximate the non-convex constraints (20b) and (20c) into convex ones. To give more detail, in [80], the authors observe that the non-convex quadratic constraint (20b) or (20c) can be seen as the intersection of two in-equality constraints. For example, (20b) can be regarded as the intersection between a convex in-equality constraint $\psi^+ \geq (\alpha + \beta)^2$ and a concave one $\psi^+ \leq (\alpha + \beta)^2$ (see an example in Fig. 17). Given this observation, [80] suggests to approximate (20b) by only considering the convex in-equality constraint $\psi^+ \geq (\alpha + \beta)^2$, along with heuristics to limit the search space

of ψ^+ to be close to the boundary of this convex constraint. Following this idea, we achieve the approximation by introducing a trust-region constraint established at the point $\psi^+ = 0$ (see the example in Fig. 17). This trust-region constraint limits the search space of ψ^+ to be close to the boundary of the convex inequality around the point $\psi^+ = 0$. Similarly, the same procedure can be also applied to approximate the non-convex constraint (20c). To summarize, following the idea described above, we can approximate the bi-linear terms $\alpha\beta$ as:

$$\alpha\beta = (\psi^+ - \psi^-), \tag{21a}$$
$$\psi^+ \geq \frac{1}{4}(\alpha + \beta)^2, \quad \psi^+ \in [0, \psi^+_{ub}] \tag{21b}$$
$$\psi^- \geq \frac{1}{4}(\alpha - \beta)^2, \quad \psi^- \in [0, \psi^-_{ub}], \tag{21c}$$

where (21b) and (21c) are the trust-region constraints placed at the points $\psi^+ = 0$ and $\psi^- = 0$, ψ^+_{ub} and ψ^-_{ub} are the upper bounds of the trust-region constraints selected by the user.

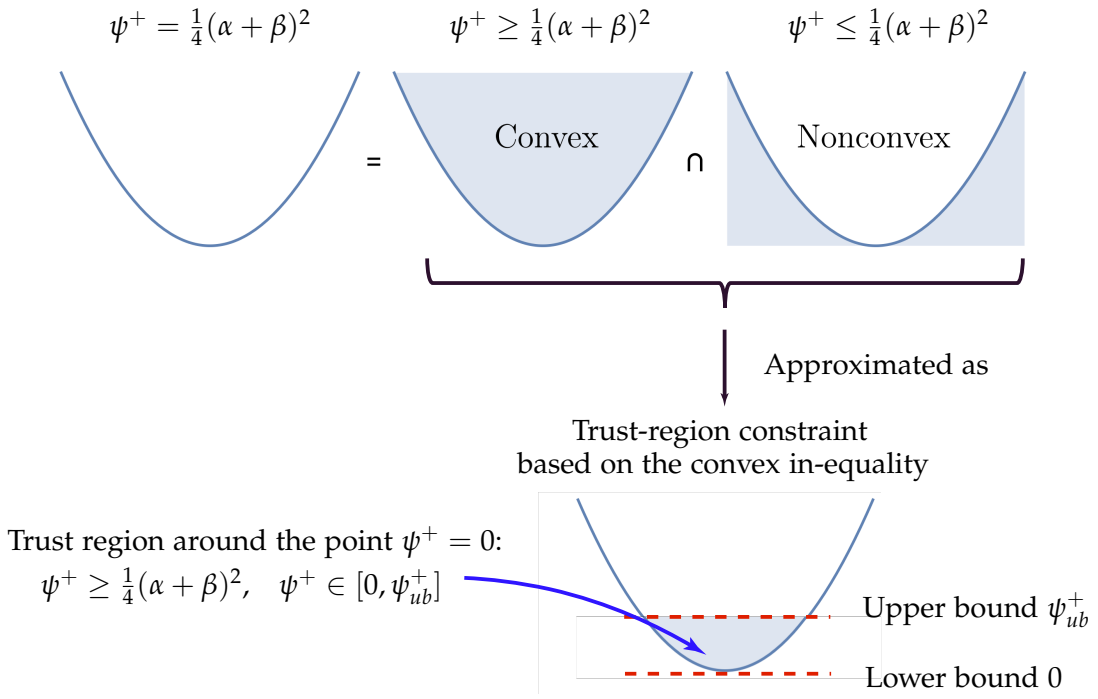


Figure 17: The non-convex quadratic equality constraint $\psi^+ = \frac{1}{4}(\alpha + \beta)^2$ can be seen as the intersection between a convex inequality constraint and a concave one. As suggested by [80], we can approximate this constraint by only considering the convex part, while at the same time trying to restrict the search space to stay near the boundary. Following this idea, we introduce a trust region constraint that limits the search space of ψ^+ to be close to the boundary of the convex inequality around the point $\psi^+ = 0$. Similarly, this approximation also applies to the other quadratic term ψ^-. The picture is adapted from [80].

In this candidate, we apply the convex approximation (21a)-(21c) to all the bi-linear terms arise from the cross product of the angular dynamics. Furthermore, in order to retain a low-dimensional model with the state vector of $\boldsymbol{x} = [\boldsymbol{c}^\top, \dot{\boldsymbol{c}}^\top]^\top$, we avoid the explicit modeling of the angular momentum $\boldsymbol{L}$. Instead, we penalize the magnitude of the auxiliary variables ψ^+ and ψ^- used for approximating the angular dynamics in the running cost. This can help us to minimize the angular momentum rate. Moreover, we also penalize large CoM acceleration in the running cost to encourage a smooth trajectory. Lastly, same as the first candidate, we also fix the phase switching timings in the prediction horizon and model the robot foot as a rectangular patch.

Compared to the centroidal dynamics model, our second candidate model has increased dimensionality due to the introduction of the auxiliary variables ψ^+ and ψ^- for approximating the angular dynamics. Nerveless, this also allows our second candidate model to be fully convex, while at the same time considering both the linear CoM dynamics as well as the angular dynamics.

5.2.3 *Candidate 3: Convex Angular Dynamics with Point Contacts*

To reduce the dimensionality of the convex approximation of the angular dynamics, we propose our third candidate model, in which we switch the rectangular foot to the point foot and apply the same modeling as described in the second candidate. As a result, the control input reduces to $\boldsymbol{u} = [\boldsymbol{f}_L^\top, \boldsymbol{f}_R^\top]^\top$, where $\boldsymbol{f}_L \in \mathbb{R}^3$ and $\boldsymbol{f}_R \in \mathbb{R}^3$ refers to the contact force vector of the left and right foot, respectively. This reduces the number of auxiliary variables (ψ^+ and ψ^-) as well as the associated trust-region constraints (21b)-(21c).

5.2.4 *Complexity Comparison of Candidate Models*

To provide an intuition of the computation complexity of these candidate models, we compare their model complexity in terms of dimensionality, and the number of non-convex and convex constraints for each knot in Table 1.

As the table indicates, the centroidal dynamics model (baseline) introduces 36 decision variables for each knot. These decision variables cover the state vector as well as the contact force vectors from each foot. Furthermore, for each knot, the centroidal model also brings 12 non-convex constraints into the TO problem, which is the main computation bottleneck that prevents online computation. In comparison, owing to the removal of the angular dynamics, our first candidate has reduced model complexity by only introducing 18 decision variables and 6 convex constraints for each knot. Although our first candidate features a simple model, our experiment

result (Chapter 7) demonstrates that ignoring the angular dynamics in the prediction horizon can lead the robot to reach ill-posed states which cause convergence failures. To add the consideration of angular dynamics while remaining convex, we present our second candidate where the angular dynamics is approximated with a convex relaxation. The convex relaxation introduces auxiliary variables and convex trust-region constraints. As a result, our second candidate brings 78 decision variables and 48 convex constraints for each knot. Although our second candidate has more decision variables, it is fully convex. Lastly, we also introduce our third candidate model, where we employ the same dynamics modeling with the second candidate, while we switch the rectangular foot to the point foot. This helps us to simplify the model complexity by reducing the number of decision variables and convex constraints to 12.

Table 1: Knot-wise model complexity of the centroidal dynamics model and the three convex relaxations.

No. of / Model	Decision variables	Non-convex Constraints	Convex Constraints
Centroidal Dynamics (Baseline)	36	12	0
Candidate 1 Linear CoM Dynamics	18	0	6
Candidate 2 Convex Angular Dynamics with Rectangular Foot	78	0	48
Candidate 3 Convex Angular Dynamics with Point Foot	12	0	12

5.3 CONCLUSION

In this chapter, we presented the principle and technical details for multi-fidelity RHP—our first approach for achieving online RHP of contact and motion plans. The key idea of multi-fidelity RHP is to relax the model complexity employed in the prediction horizon. To this end, we present three candidate convex relaxations of the centroidal model.

In the literature, a similar framework to the one we propose was introduced by [59]. The authors present a MPC framework based on DDP that combines whole-body dynamics and a non-convex model with reduced order (single-rigid body model [105, 109]) in a single formulation. Successful MPC of 2D quadrupedal locomotion and

humanoid running has been demonstrated on flat surfaces. In contrast, the emphasis of our work is RHP of centroidal trajectories for uneven-terrain locomotion of humanoid robots. This problem requires careful selection of contact locations and timings to modulate the centroidal momenta. In this regard, the simplified model applied in the prediction horizon needs to be carefully designed, as the quality of the model can significantly affect the performance of the framework. Further, instead of searching for non-convex models with reduced order, we focus on finding appropriate convex relaxations for the prediction horizon.

To evaluate the computation performance of our multi-fidelity RHP approach, we will present rigorous experiments over a set of multi-contact scenarios in Chapter 7.

6

LEARNING TO GUIDE ONLINE MULTI-CONTACT RECEDING HORIZON PLANNING

In this chapter, we present the technical details of our second approach for achieving online Receding Horizon Planning (RHP) of contact and motion plans on uneven terrain. We call this approach as Locally-Guided Receding Horizon Planning (LG-RHP).

In comparison to traditional RHP and multi-fidelity RHP which approximate the value function by computing the trajectory in the prediction horizon, our locally-guided RHP explores the possibility of learning a value function model from past experiences. Given this learned value function model, we can use it as the terminal cost of the RHP problem and shorten the planning horizon only to cover the execution horizon. This allows us to significantly reduce the complexity of the RHP problem and thus improve the computation efficiency.

In this chapter, we organize the contents as follows. First, in Section 6.1, we provide an approach overview to describe the core idea of our locally-guided RHP. Then, in Section 6.2, we elaborate on the technical detail of the approach. Next, in Section 6.3, we discuss the reasoning behind the design choices we made in our approach. Lastly, we conclude this chapter with Section 6.4.

6.1 APPROACH OVERVIEW

In Chapter 5, we have introduced the technical details of multi-fidelity RHP, our first approach for achieving online RHP of contact and motion plans. This approach shares similarities with the traditional RHP in the sense that they both approximate the value function by planning trajectories in the prediction horizon (see an illustration in Fig. 18-a). Nevertheless, unlike the traditional RHP which computes the prediction horizon with an accurate dynamics model, our multi-fidelity RHP chooses to plan the prediction horizon with a convex relaxed model. This makes our multi-fidelity RHP have reduced computation complexity. However, it is worth to note that, approximating the value function with the prediction horizon always brings an extra computation burden in addition to the planning of the execution horizon. Even if we plan the prediction horizon with simplified models, such extra computation burden can still become dominant when a long planning horizon is considered, which can in turn make our multi-fidelity RHP struggle to compute online.

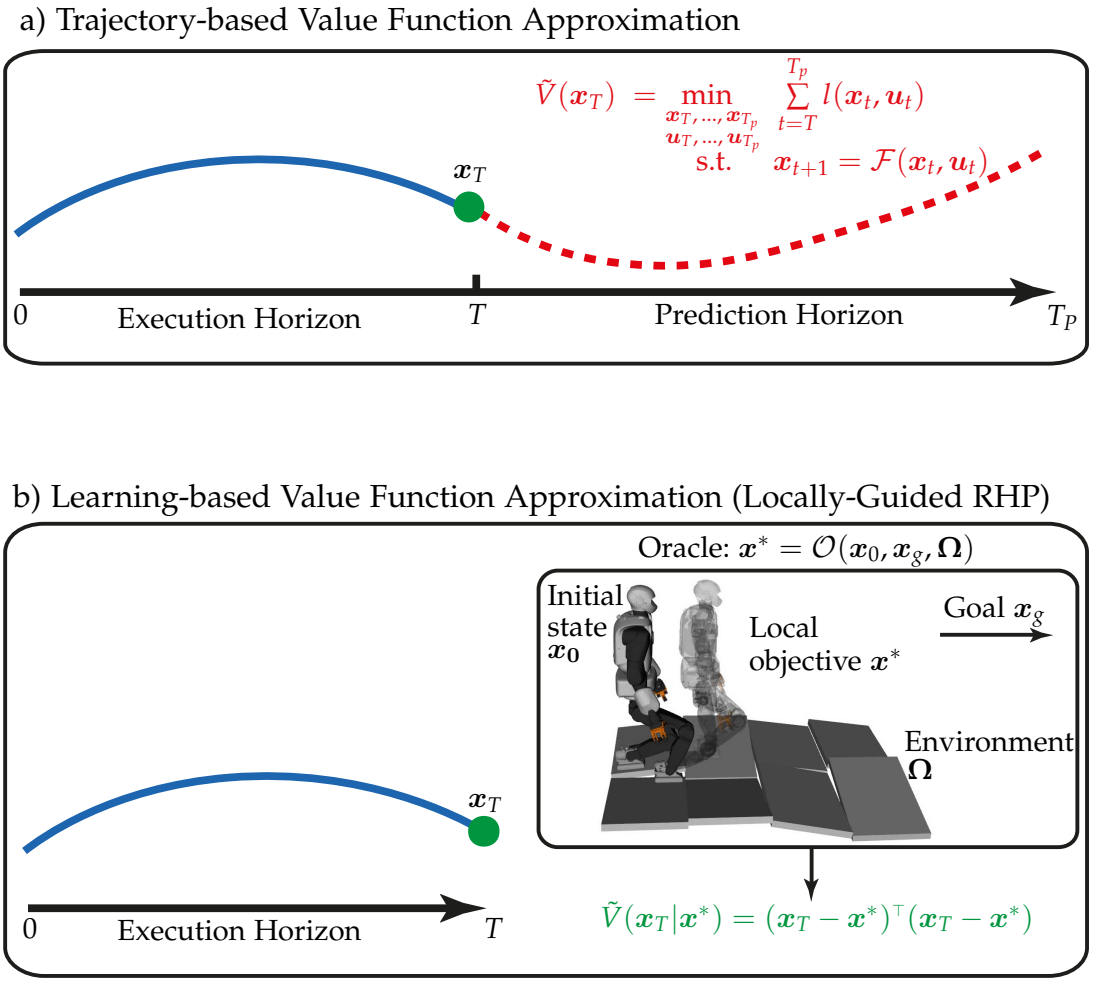


Figure 18: a) Trajectory-based value function approximation methods model the value function by computing trajectories in the prediction horizon while considering a robot dynamics model that governs the motion. Nevertheless, these trajectory-based value function approximation methods bring extra computation burden, which may prevent online computation; b) Our learning-based approach (locally-guided RHP) learns an oracle to predict local objectives $\boldsymbol{x}^*$ based on the initial state $\boldsymbol{x}_0$, the final goal $\boldsymbol{x}_g$, and the environment model Ω. Given the predicted local objectives, we construct local value functions $\tilde{V}(\boldsymbol{x}_T|\boldsymbol{x}^*)$ to guide a short-horizon RHP that only covers the execution horizon.

To further accelerate the computation speed, in this chapter, we explore the possibility of approximating the value function without computing the prediction horizon. To achieve this goal, one of the options is to approximate the value function with a learned model. Nevertheless, learning a value function for the contact and motion planning problem can be challenging. The main difficulty comes from the fact that the value function for this problem is defined in a coupled state-environment space, which requires a flexible parameterization that can capture the landscape changes of the

value function with respect to environment variations [25]. To circumvent this issue, we propose to learn an oracle that can predict local objectives—intermediate goal states towards the completion of a given task—based on the current robot state, goal position, and the environment model. These local objectives are then used for constructing local value functions that guide a short-horizon TO to plan the execution horizon (see an illustration in Fig. 18-b). We call this approach as locally-guided Receding Horizon Planning (RHP). To obtain the oracle, we take a supervised learning approach to learn the oracle from the expert knowledge offline computed from traditional RHP.

Next, we describe the technical details of our locally-guided RHP approach.

6.2 TECHNICAL APPROACH

In this section, we describe the technical details of locally-guided RHP in the context of computing contact and motion plans. More specifically, in Section 6.2.1, we firstly introduce the oracle formulation along with the definition of the associate variables (See Fig. 19-a). Then, in Section 6.2.2, we present our supervised learning approach for obtaining the oracle. Lastly, in Section 6.2.3, we explain how to construct a local value function based on the output of the oracle and interface to a short-horizon TO.

6.2.1 *Oracle Formulation*

As mentioned in Section 4.3, we define that the execution horizon always covers the motion plan for making one step. As a result, our oracle is designed to predict the local objective for making one step. We define the oracle $\mathcal{O}$ as:

$$\boldsymbol{x}^*, \boldsymbol{p}^*, \boldsymbol{\mathcal{T}}^* = \mathcal{O}(\delta_{l/r}, \boldsymbol{x}_0, \boldsymbol{p}_0, \boldsymbol{\Omega}, \boldsymbol{x}_g). \tag{22}$$

Variable Definitions:

The output of the oracle describes the goal configuration for making a step, which includes:

- $\boldsymbol{x}^*$: the target CoM state after making one step.
- $\boldsymbol{p}^*$: the target contact location for the swing foot to reach.
- $\boldsymbol{\mathcal{T}}^* = \{\tilde{t}^1, \tilde{t}^2, \tilde{t}^3\}$: the target phase switching timings for the three phases that compose of the step.

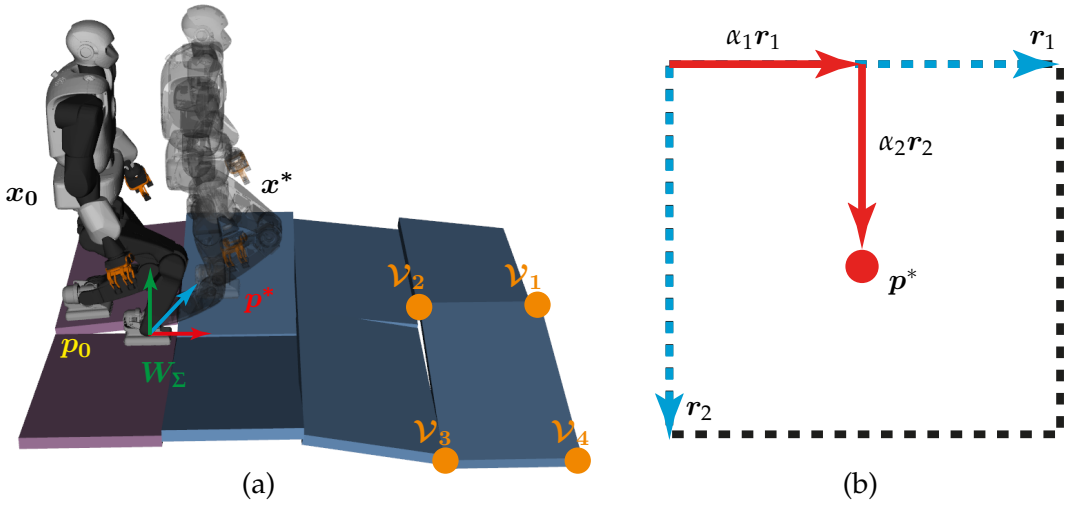


Figure 19: (a) Definition of the oracle variables. x_0 is the initial CoM state, and p_0 is the initial contact location of the swing foot. x^* is the target CoM state, and p^* is the target contact location of the swing foot. The purple patches are the initial contact surfaces for the left and the right foot, while the blue patches are the contact surfaces for future n steps. We model each contact surface with its four vertices $\mathcal{V}_i$. We define all spatial terms in the contact foot frame W_Σ established at the stationary foot (non-swing foot), while the orientation of the frame is aligned to the surface in contact. (b) The target contact location p^* is represented as the vector sum of $\alpha_1 r_1$ and $\alpha_2 r_2$, which scale along the borders of the contact surface $r_1, r_2 \in \mathbb{R}^3$, with the proportion defined by $\alpha_1, \alpha_2 \in [0, 1]$.

To predict the target robot configuration, the oracle needs to take into account the following inputs:

- $\delta_{l/r} \in \{0, 1\}$: the swing foot indicator telling which foot (left/right) is going to re-position its location. We use 0 to denote the left foot is going to swing to a new location, while we use 1 to indicate the right foot is going to move to a new place.
- x_0: the initial CoM state.
- p_0: the initial contact location of the swing foot.
- $\Omega = \{\mathcal{S}_{l0}, \mathcal{S}_{r0}, \mathcal{S}_1, ..., \mathcal{S}_n\}$: a local preview of the environment model. We define $\mathcal{S}_{l0}$, $\mathcal{S}_{r0}$ as the contact surfaces that the left and right feet initially stand upon, $\mathcal{S}_1, ..., \mathcal{S}_n$ as the contact surfaces that future n steps will land on. Each contact

surface is represented by its four vertices: $\mathcal{S} = \{\mathcal{V}_1, ..., \mathcal{V}_4\}$, where $\mathcal{V}_i \in \mathbb{R}^3, i \in \{1, ..., 4\}$ is the 3-D location of the i-th vertex.

- $\boldsymbol{x}_g$: a final goal state which is a fixed goal location in our case.

We illustrate the definition of the oracle variables in Fig. 19-a. Furthermore, as the figure indicates, we define the spatial quantities such as the CoM states, contact locations, and the environment model in the so-called contact-foot frame $\boldsymbol{W}_\Sigma$. This frame locates at the position of the stationary foot (the non-swing foot), while having the same orientation with respect to the surface in contact.

Additionally, we introduce an on-surface parameterization for the target contact location (Fig. 19-b), which transforms the 3-D target contact location as the sum of two vectors:

$$\boldsymbol{p}^* = \alpha_1 \boldsymbol{r_1} + \alpha_2 \boldsymbol{r_2}, \tag{23}$$

scaling along vectors of the surface borders $\boldsymbol{r}_1, \boldsymbol{r}_2 \in \mathbb{R}^3$, and we predict the scaling factors $\alpha_1, \alpha_2 \in [0, 1]$.

6.2.2 *Learning the Oracle*

To model the oracle function, we choose a Neural Network (NN) of 4 hidden layers. We define each layer has 256 neurons with ReLu activation functions. The NN is implemented with Tensorflow framework [1].

When using the oracle to guide the planning of the execution horizon, one of the issues is that the oracle can have prediction errors. Such prediction error typically comes from imperfect fitting or insufficient data coverage. These prediction errors can lead our locally-guided RHP to arrive at ill-posed states, from which the TO can fail to converge.

To mitigate this issue, we employ an incremental training scheme. The key idea of our approach is to improve the prediction accuracy by incrementally adding data points to demonstrate recovery actions from the states that cause convergence failures.

To give more detail, as illustrated in Fig. 20, in each training iteration i, we train an oracle $\mathcal{O}_i$ based on the current data-set:

$$\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_1^* \cup ... \cup \mathcal{D}_{i-1}^*, \tag{24}$$

where $\mathcal{D}_0$ is the initial data-set, and $\mathcal{D}_1^* \cup ... \cup \mathcal{D}_{i-1}^*$ are the augmented data-sets obtained from previous training iterations. The initial data set $\mathcal{D}_0$ is achieved by rolling out the traditional RHP with a 3-step prediction horizon over a set of randomly sampled environments and then extracting the data points from the execution horizon of each cycle. The reason why we choose traditional RHP with a 3-step prediction horizon for data generation is that we find a longer lookahead does not improve the quality of the motion (cost) while increasing the computation time. For computing the augmented data set $\mathcal{D}_i^*$, we firstly use locally-guided RHP with the currently trained oracle $\mathcal{O}_i$ to plan trajectories in a RHP fashion on the previously sampled environments. For the failed roll-outs, we use the traditional RHP to compute recovery actions, which starts from 1 to 3 cycles before the locally-guided RHP fails, until the cycle converges back to the ground-truth trajectory (roll-out of traditional RHP on the same terrain). We repeat the process until there is no further improvement in the convergence rate.

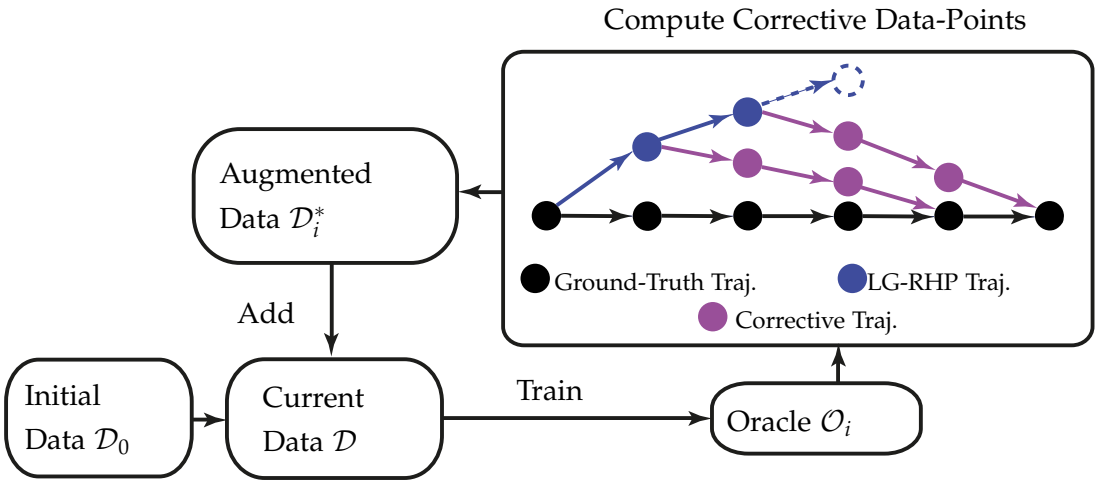


Figure 20: The concept of the incremental training scheme. In each training iteration i, we train an oracle $\mathcal{O}_i$ based on the data-set $\mathcal{D}$ that aggregates the initial data-set $\mathcal{D}_0$ and the augmented data sets $\mathcal{D}_i^*$. The augmented data set contains recovery actions (purple), starting from 1-3 cycles before locally-guided RHP fails (dashed blue circle) until the cycle converges back to the ground-truth trajectory (computed by the traditional RHP).

6.2.3 *Interfacing to the Short-horizon RHP*

To guide the short-horizon RHP which only plans the execution horizon, we adapt the short-horizon version of the TO problem (11a) with the following changes.

First, we replace the terminal cost with the following quadratic term:

$$(\boldsymbol{x}_T - \boldsymbol{x}^*)^\top (\boldsymbol{x}_T - \boldsymbol{x}^*) + (\boldsymbol{p}^1 - \boldsymbol{p}^*)^\top (\boldsymbol{p}^1 - \boldsymbol{p}^*), \tag{25}$$

This cost function encourages the terminal state $\boldsymbol{x}_T$ and the contact location $\boldsymbol{p}^1$ to approach the predicted targets $\boldsymbol{x}^*$ and $\boldsymbol{p}^*$.

Second, we introduce constraints to narrow down the search space of phase switching timings t^i around their predicted values:

$$(1 - \epsilon)\tilde{t}^q \leq t^q \leq (1 + \epsilon)\tilde{t}^q, \tag{26}$$

where ϵ is a user-defined slack. This is because we empirically find that reducing the search space of the phase switching timings can result in more efficient computation than using cost terms to bias their decisions.

6.3 DISCUSSION ON THE DESIGN CHOICES

In Section 6.2, we have presented the technical details of our locally-guided RHP approach. As we can notice, the development of locally-guided RHP involves the decisions on the following design choices: 1) choosing the machine learning model for representing the oracle; and 2) finding an effective data augmentation technique. In this section, we will discuss the reasoning behind these design choices.

6.3.1 *Choosing the Machine Learning Model for Representing the Oracle*

One of the design choices involved in our locally-guided RHP is to choose a machine learning model to represent the oracle. Generally speaking, the problem of learning the oracle can be seen as a regression task. To address this regression task, there are several machine learning techniques that can be considered.

For instance, one of the potential options for implementing the oracle is to use the k-Nearest Neighbor (k-NN) method [21]. The k-NN is a non-parametric method that makes predictions based on the average of the k-nearest data points in the training data set. However, due to the following reasons, we did not choose to use k-NN to model the oracle. First, in order to make the prediction, k-NN necessarily requires the selection of k data points that are similar to the input vector. However, in our case, the input vector of the oracle involves both the robot configuration (CoM state,

contact location) and the environment model, and it is unclear on how to measure the similarity in such complex input space. Second, the choices of the hyper-parameter k which specifies the number of neighbors to be considered can largely impact the prediction accuracy of k-NN, and finding a correct k can be a tedious task. Lastly, the k-NN can also struggle to achieve fast computation when searching for neighbors in a large data set.

Alternatively, another option for modeling the oracle can be using the Gaussian Process Regression (GPR) [86]. GPR is also a non-parametric method while it uses Gaussian Process to capture the relationship between input and output. However, GPR also requires hyper-parameter tuning, such as choosing the kernel function and specifying the noise level of the data. These hyper-parameters can affect the prediction accuracy of the GPR, and optimizing these hyper-parameters can be non-trivial. Additionally, GPR also does not scale well on the large data set, as it requires the inversion of a large covariance matrix, which can be computationally challenging.

Eventually, to model the oracle, we decide to use a Neural Network (NN) model—a parametric model composed of multiple layers of inter-connected neurons. The neurons in each layer connect to the neurons in the next layer through a set of weights. These weights are the parameters that shape the relationship between the input and the output, and they can be trained during the learning process. Once the training is finished, the NN can make predictions by feeding the input vector into the model and then passing it through the hidden layers to the output layer. In theory, NN model is very flexible and can model any continuous functions [42], and it can learn from a large amount of data. The computation load of the neural network typically depends on the size of the model, i.e., the number of hidden layers, and the number of neurons in each layer. It is believed that the neural network can become more flexible if we choose a large number of hidden layers and neurons. In our case, in order to achieve efficient computation, we choose to build a small-sized NN model which contains only 4 hidden layers with 256 neurons in each layer. We tested that the computation of this model can only take 1ms. Furthermore, we also find that this architecture is flexible enough to model the oracle, and increasing the number of hidden layers and neurons does not improve much on the prediction accuracy.

6.3.2 *Finding Effective Data Augmentation Techniques*

As mentioned in Section 6.2.2, due to imperfect fitting, the oracle may have prediction errors. These prediction errors can cause the locally-guided RHP to diverge from the ground-truth trajectories demonstrated by the traditional RHP, and reach the states that are un-explored by the training data set. When reaching these unseen states, the prediction accuracy of the oracle can drop dramatically, and this can lead the robot to eventually arrive at an ill-posed state that can cause convergence failures (see an example in Fig. 21-a). To mitigate this issue, we need data augmentation techniques that can expand the data coverage of the training data set. To achieve this goal, we have attempted several data augmentation techniques, and we eventually converge to the method explained in Section 6.2.2. In this section, we will present our exploration of the data augmentation techniques and discuss our insights.

As Fig. 21-b illustrates, to expand the data coverage, the first data augmentation technique we have tried is to sample new robot states around the ground-truth trajectories demonstrated by the traditional RHP. These new robot states can be generated by injecting uniformly sampled noise into the robot state vector of the existing training data points. We assume that these newly sampled robot states do not diverge too much away from the ground-truth trajectories, and they can return back to the ground-truth trajectories with one RHP cycle. This means that the output vector of these newly sampled robot states will be the same as the existing training data points. To implement this data augmentation strategy, we can add the noise into the input vectors of the training data set during the training process, while at the same time maintaining the output vectors unchanged. Nevertheless, during our test, we find that this data augmentation technique does not improve the convergence rate of locally-guided RHP. We notice that, during the roll-out of locally-guided RHP, the occurrence of the unseen states depends on the oracle trained at hand. Furthermore, these unseen states can have large deviations from ground-truth trajectories. We suspect that randomly sampling the new robot states around ground-truth trajectories is unlikely to cover the unseen states induced by the prediction error of the oracle.

To have a better data coverage, we attempt another data augmentation technique called DAGGER (Dataset Aggregation) [84]. As Fig. 21-c illustrates, DAGGER is an on-policy data augmentation strategy which generates the new data points by demonstrating recovery actions starting from all the states appeared from the roll-out of the learned policy, until return back to the ground truth trajectories. These can include both the states that are close to the ground-truth trajectories, and the ones that cause convergence failures (typically, they are away from the ground-truth trajectories). Throughout our test, we observe that the DAGGER can improve the convergence rate

of locally-guided RHP after the first iteration of the data augmentation procedure. However, if we continue to add new data points with more iterations of DAGGER, we find that the convergence rate of locally-guided RHP will decrease. This is because the majority of the states that appeared from the roll-out of locally-guided RHP are close to the ground-truth trajectories. Computing recovery actions from these states will result in more data points that are similar to the ones already exist in the training data set. Adding these data points does not help to expand the data coverage, but in the meantime leads to an un-balanced data set: the data points around the ground-truth trajectories can become more dominant and appear more frequently during the training process, while the data points telling how to recover from the states that cause convergence failures will have less weight in the training data set and appear less frequently during the training process. Training the Neural Network (NN) with this in-balance data set can dramatically affect the prediction accuracy of the oracle.

To expand the data coverage in a more targeted fashion, we propose our data augmentation technique that only adds recovery actions from the states that cause convergence failures (see the illustration in Fig. 21-d). We decide that these recovery actions should start from 1 to 3 cycles before the locally-guided RHP fails to converge. We make this decision because we find that the robot usually has large state deviations from the ground-truth trajectories at 1 to 3 cycles before the convergence failures. As we will present in Section 7.4.4, our data augmentation strategy can continuously improve the convergence rate of the locally-guided RHP. Nevertheless, we also notice that it is hard to achieve a 100% convergence rate of locally-guided RHP. We believe further investigation on how to systematically generate the data set is important.

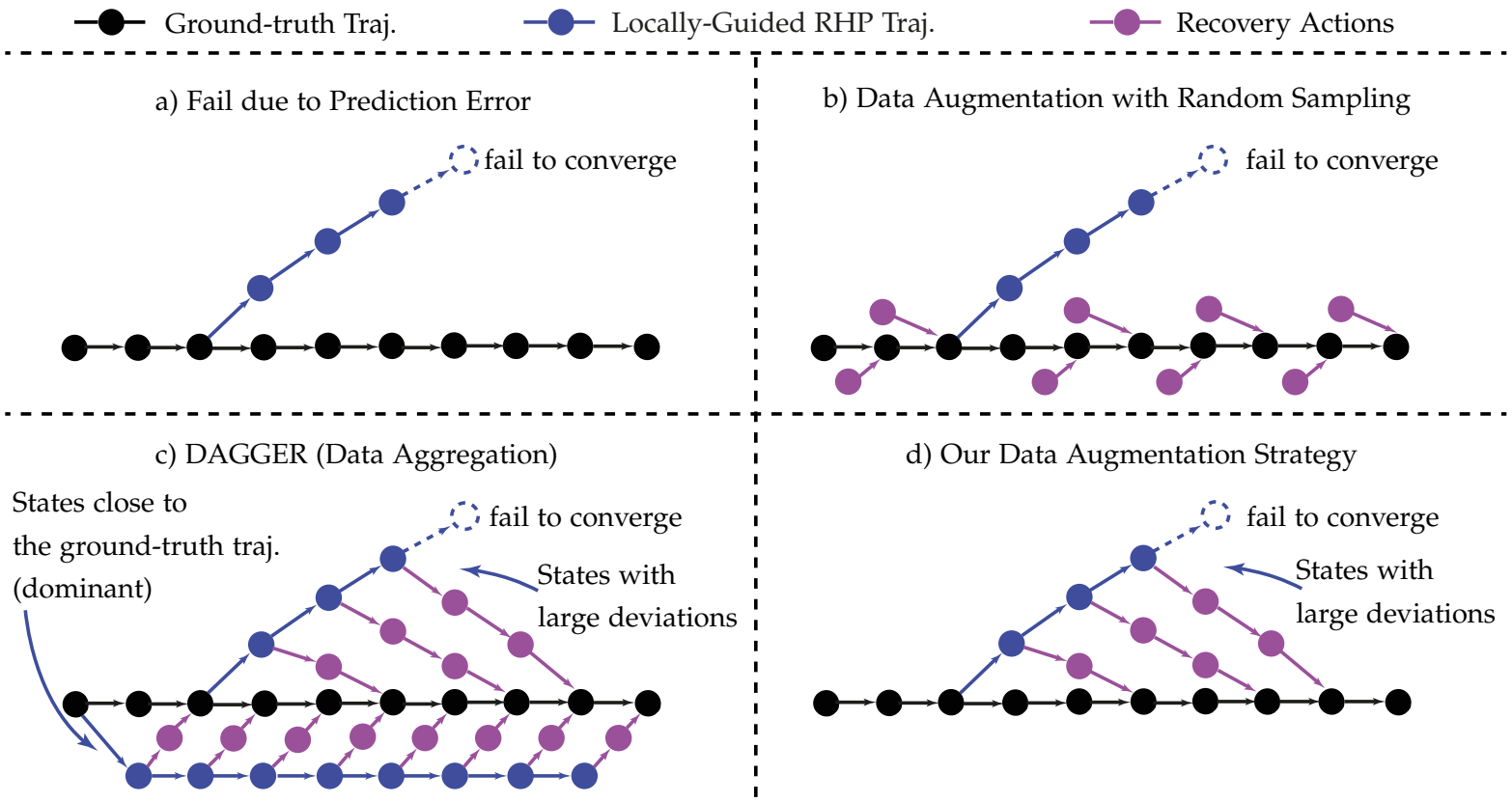


Figure 21: a) The prediction error of the oracle can lead locally-guided RHP to diverge from the ground-truth trajectory (demonstrated by the traditional RHP), and eventually cause convergence failures. To improve the prediction accuracy, we need to augment the data set. b) We can augment the data set by randomly sampling states around the ground-truth trajectories. However, we find this method is hard to cover the unseen states appeared from the roll-out of locally-guided RHP. c) Data Aggregation (DAGGER) [84] suggests to add recovery actions starting from all the states appeared from the roll-out of the learned policy. However, we find that, keep adding data points with DAGGER can result in a decrease of the convergence rate. This is because the majority of the states appeared from the roll-out of locally-guided RHP are the ones close to ground-truth trajectory. Keep adding recovery actions from these states will lead to an unbalanced data set, i.e., the data points that are similar to the ground truth trajectory will become dominant, and the oracle can struggle to learn how to recover from large state deviations, as these actions appear less frequently in the data set. d) We propose a more targeted data augmentation strategy where we only focus on adding recovery actions starting from the states that cause convergence failures, i.e., 1 to 3 cycles before failure, as we find these states exhibit large deviations from the ground-truth trajectory.

6.4 CONCLUSION

In this chapter, we presented the technical detail of locally-guided RHP—our second approach to achieve online RHP of contact and motion plans. Differs from traditional RHP and multi-fidelity RHP which model the value function by a trajectory that looks into the future, our locally-guided RHP features a learned value function model. This helps us to reduce the planning horizon. However, learning a value function for the contact and motion planning problem can be challenging. The major difficulty is the landscape of the value function can vary with respect to environment changes. To capture such landscape changes, we need a flexible parameterization that can represent the value function in the coupled state-environment space. To tackle this issue, we proposed the idea of learning an oracle to predict local objectives for completing a given task. We then construct local value functions based on these local objectives to guide the planning of the short-horizon RHP. Following this idea, we presented the formulation and associate variable definitions for the oracle in the context of planning contact and motion plans for a legged robot to traverse uneven terrain, and we introduce an incremental training scheme that can iteratively improve the prediction accuracy of the oracle. Lastly, we also discussed the reasoning behind the design choices we made in our locally-guided RHP approach, i.e., choosing the machine learning model for representing the oracle and finding an effective data augmentation strategy. In the next chapter, we will present a simulation study to evaluate the computation performance of our locally-guided RHP, in comparison to the multi-fidelity RHP and the traditional RHP.

7

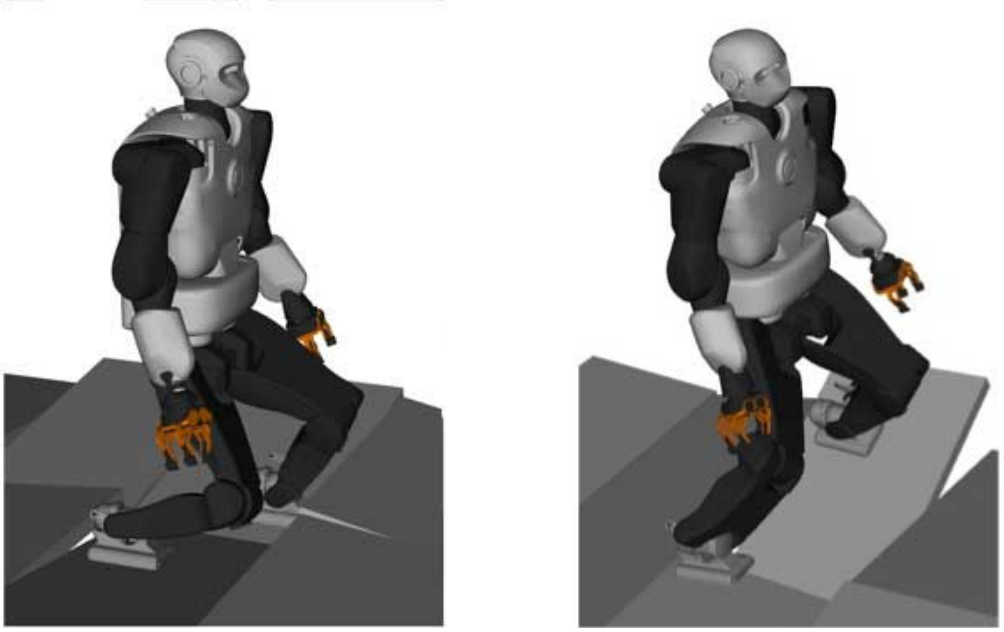
SIMULATION STUDY OF OUR ONLINE MULTI-CONTACT RECEDING HORIZON PLANNING APPROACHES

In the previous chapters, we have introduced the principle and technical details of our proposed methods for achieving online RHP of contact and motion plans, namely multi-fidelity RHP and locally-guided RHP. The key idea of these two RHP approaches is to reduce the computation complexity of the TO problem by finding efficient approximations of the value function. To this end, our multi-fidelity RHP relaxed the model accuracy in the prediction horizon, while our locally-guided RHP focuses on shortening the planning horizon by learning a value function model.

In this chapter, we carry out a simulation study to evaluate the computation performance of our proposed RHP approaches over a set of multi-contact scenarios. More specifically, we firstly explain the experiment setup in Section 7.1, and then we describe the implementation details in Section 7.2. Afterward, we evaluate each RHP framework with the following two case studies: 1) the moderate slope case (Section 7.3), and 2) the large slope case (Section 7.4). Additionally, in Section 7.4.4, we also show that our incremental training scheme (Section 6.2.2) can successfully improve the perdition accuracy of the oracle employed in locally-guided RHP. Lastly, we discuss the advantages and disadvantages of each RHP framework in Section 7.5 and conclude the chapter in Section 7.6. Throughout our simulation studies, we find that our multi-fidelity RHP can accelerate the computation speed by relaxing the model accuracy in the prediction horizon. However, the relaxation cannot be arbitrary. Moreover, owing to the shortened planning horizon, we find the locally-guided RHP demonstrates the best computation efficiency.

7.1 EXPERIMENT SETUP

We consider the following two types of terrains: 1) moderate slope terrain (Fig. 22-a) and 2) large slope terrain (Fig. 22-b). Planning multi-contact motions on these terrains can be challenging. The key issue is that the admissible contact force is limited by the orientation of the surface in contact. As a result, in order to find a feasible momentum trajectory of the CoM, the planning algorithm has to carefully select the contact locations and the timings [80].



a) Moderate Slope (5-12 degrees)

b) Large Slope (17-26 degrees)

Figure 22: Our experiments consider the following two types of terrain: a) the moderate slope terrain (5-12 degrees), and b) the large slope terrain (17-26 degrees). The large slope terrain is more challenging as the robot cannot maintain static stability when traversing the large slope. This is because the friction cone does not contain the contact force vectors that can cancel the gravity.

On these terrains, we use each RHP framework to offline plan centroidal trajectories of the humanoid robot Talos [90] in a RHP fashion. To give more detail, we consider a RHP loop where each planning cycle aims to compute the motion plan to be executed for the next cycle. Under the assumption that the controller can track the planned motion without having large deviations, we enforce that the motion plan of the next cycle always starts from the terminal state of the execution horizon planned for the current cycle.

To highlight the computation benefit of our proposed RHP frameworks, we consider an online setting where we impose a computation time limit in each cycle. To give more detail, we denote a cycle achieves online computation if the TO converges within the time budget—the duration of the motion (execution horizon) to be executed in the current cycle. In the case of the TO failing to converge within the time budget, we still leave the TO to compute until convergence unless no solutions are found (fail to converge). We test all the RHP frameworks on the terrains that are unseen during

the training of the locally-guided RHP, and we refer to the trial on each terrain as an episode. To validate the dynamic feasibility of the planned trajectories, we track them using a whole-body inverse dynamics controller [27] in simulation.

7.2 IMPLEMENTATION DETAILS

In this thesis, all the TO problems are modeled using Python, and we solve them with the interior-point method of KNITRO 10.10 [12]. Furthermore, we employ the automatic differentiation framework CasADi [3] to provide the gradients and the Hessians to the optimization solver. All the computations are carried out on a desktop with an Intel i9 CPU (3.6GHz) and 64GB memory.

For locally-guided RHP, we train separate oracles for the two types of the terrains. This is because we find the data distributions of these two types of the terrains have different modalities, i.e., when traversing the large slopes, the robot tends to exhibit larger momentum variations than walking on the moderate slopes. Mixing these data points can lead to a discontinuous and unbalanced data set, on which the single Neural Network model can struggle to interpolate. In Section 7.5, we discuss the potential options that can generalize across these two modalities. In our experiments, the locally-guided RHP employs the oracle with the best prediction accuracy.

7.3 CASE STUDY 1 (CS1): MODERATE SLOPE

In this section, we present the experiment result of our first case study: walking on moderate slopes (Fig. 22-a). Although we can quickly find quasi-static motions for this type of terrain (this is because the force vectors from the friction cone associated with each contact surface can cancel the gravity [87]), we are interested in planning dynamic walking motions using TO. This provides us with a unified approach to handle non-quasi-static cases, such as the large slope terrain. Furthermore, walking dynamically can also allow the robot to complete the locomotion task more efficiently. In this case study, we set the slack of the phase switching timing constraint (26) for locally-guided RHP as $\epsilon = 0.6$. We find this can enlarge the search space of the trajectories considered by locally-guided RHP, and thus increase the chance of finding a solution, while in the meantime we do not sacrifice much computation time for locally-guided RHP.

We evaluate the performance of each RHP framework based on episodic success rate and cycle-wise success rate in both the offline and online settings. We declare an episode is successful if the chosen RHP framework can compute the motion plan for all the cycles within the episode. In this case study, we define that each episode

contains a maximum number of 28 cycles. We list the statistics of the success rates in Table 2.

7.3.1 *Computation Performance of the Baseline (Traditional RHP)*

As Table 2 indicates, in the offline mode where we consider an unlimited time budget, the baseline can achieve a 100% episodic success rate on the moderate slope terrain with only 1-step prediction horizon. This means the baseline can successfully find solutions for all the cycles (100% cycle-wise success rate). However, due to the non-convex nature of the centroidal dynamics constraint, the baseline has nearly half of the cycles (48.69%) fail to converge online, i.e., the computation exceeds the motion duration (time budget) of the previous cycle.

7.3.2 *Computation Performance of Multi-Fidelity RHP*

In Table 2, the experiment result of multi-fidelity RHP demonstrates that we can *improve the computation efficiency by trading off the model accuracy in the prediction horizon. However, the trade-off cannot be arbitrary.* For instance, although our first multi-fidelity RHP candidate features the simplest model in the prediction horizon (only linear CoM dynamics is considered), it always fails to complete an episode after a few cycles, no matter how many steps lookahead we assign to the prediction horizon. This suggests that considering the angular dynamics in the prediction horizon is critical, as the angular dynamics can affect the accuracy of the value function approximated by the prediction horizon. Indeed, despite we consider convex relaxed angular dynamics constraints in our second and third multi-fidelity RHP candidate, both of them can achieve an offline episodic success rate of 72.5% to 79.49% with only 1-step prediction horizon. Owing to the relaxed dynamics model in the prediction horizon, our second multi-fidelity RHP candidate can achieve 69.22% of the cycles converging online, which outperforms the baseline (51.31% cycles computed online). This demonstrates that reducing the non-convexity of the TO problem can improve computation efficiency. Moreover, as our third multi-fidelity RHP reduces the dimensionality of the convex relaxation by switching from rectangular foot to point foot, it further improves computation efficiency and increases the online cycles-wise success rate to 75.11%.

To highlight the computation efficiency of our multi-fidelity RHP approach, we compare the average computation time of our second and third multi-fidelity RHP candidate against the baseline in Table 3. As the table indicates, when considering 1-step prediction horizon, our second and third multi-fidelity RHP candidate can achieve averagely 2.3x to 2.6x computation time gain than the baseline. Furthermore,

as indicated in Table 3, our multi-fidelity RHP candidates often have less time budget to spend, since they tend to generate motions plans with shorter duration. Nevertheless, owing to the improved computation efficiency, our multi-fidelity RHP candidates still achieve a higher online convergence rate than the baseline (see Table 2).

On the other hand, we also notice that our second and third multi-fidelity RHP have the risk of failing to converge. For example, when considering 1-step prediction horizon, the second and the third multi-fidelity RHP candidate can fail during 20.52% to 27.50% of the episodes due to convergence issues. Although we can improve the episodic convergence rate by extending the length of the prediction horizon, this can increase the dimensionality of the TO problem and hinders online computation. For instance, when considering the 3-step prediction horizon, both of our second and third multi-fidelity RHP candidates can achieve a high episodic success rate (97%) close to the baseline. Nevertheless, due to the increased TO dimensionality, our second multi-fidelity RHP candidate fails to achieve online computation for 94.25% cycles. Similarly, although our third multi-fidelity RHP candidate employs a convex model with reduced order in the prediction horizon, it still has 50.14% cycles that fail to converge online when we consider a 3-step prediction horizon.

In Fig. 23a and Fig. 23b, we show the snapshots of the simulation results of our second and third multi-fidelity RHP candidate with 1-step prediction horizon.

7.3.3 *Computation Performance of Locally-Guided RHP*

Compared to the baseline and multi-fidelity RHP, we show that *locally-guided RHP achieves the fastest computation speed*, where 98.63% of the cycles converge online. Furthermore, owing to the fast computation, our locally-guided RHP can maintain online computation for 67.57% episodes. In contrast, the baseline and the multi-fidelity RHP struggle to achieve online computation consecutively for a complete episode. Additionally, as Table 4 indicates, our locally-guided RHP only consumes on average 19% of the time budget. This suggests the potential of using locally-guided RHP in real-world robot control, as the remaining time budget can be allocated to the overheads, e.g., trajectory interpolation and data transmission. However, due to the prediction error of the oracle, locally-guided RHP also has the chance to fail to converge. For example, our locally-guided RHP failed 24.43% episodes as the oracle directs the robot towards ill-posed states from which the TO struggles to find a solution.

In Fig. 24, we show a sequence of the simulation snapshots for our locally-guided RHP.

Table 2: Computation Performance for the Moderate Slope Terrain (CS1).

Method		Episodic Success Rate				Cycle-wise Success Rate			
		Success (Offline)	Success (Online)	Time Out	Fail to Converge	Success (Offline)	Success (Online)	Time Out	Fail to Converge
Baseline	1-Step PH	100.0%	0.0%	100.0%	0.0%	100.0%	51.31%	48.69%	0.0%
MF-RHP 1 (CoM)	1 to 3-Step PH	0.0%	-	-	-	-	-	-	-
MF-RHP 2 (Rectangle)	1-Step PH	72.50%	0.0%	72.50%	27.50%	98.83%	69.22%	29.61%	1.17%
	2-Step PH	76.32%	0.0%	76.32%	23.68%	99.05%	44.88%	54.17%	0.95%
	3-Step PH	97.37%	0.0%	97.37%	2.63%	99.91%	5.66%	94.25%	0.09%
MF-RHP 3 (Point)	1-Step PH	79.49%	0.0%	79.49%	20.51%	99.16%	75.11%	24.90%	0.84%
	2-Step PH	83.78%	0.0%	83.78%	16.22%	99.38%	67.08%	32.30%	0.62%
	3-Step PH	97.5%	0.0%	97.5%	2.5%	99.91%	49.78%	50.14%	0.09%
LG-RHP	-	75.68%	67.57%	8.11%	24.32%	99.05%	98.63%	0.42%	0.95%

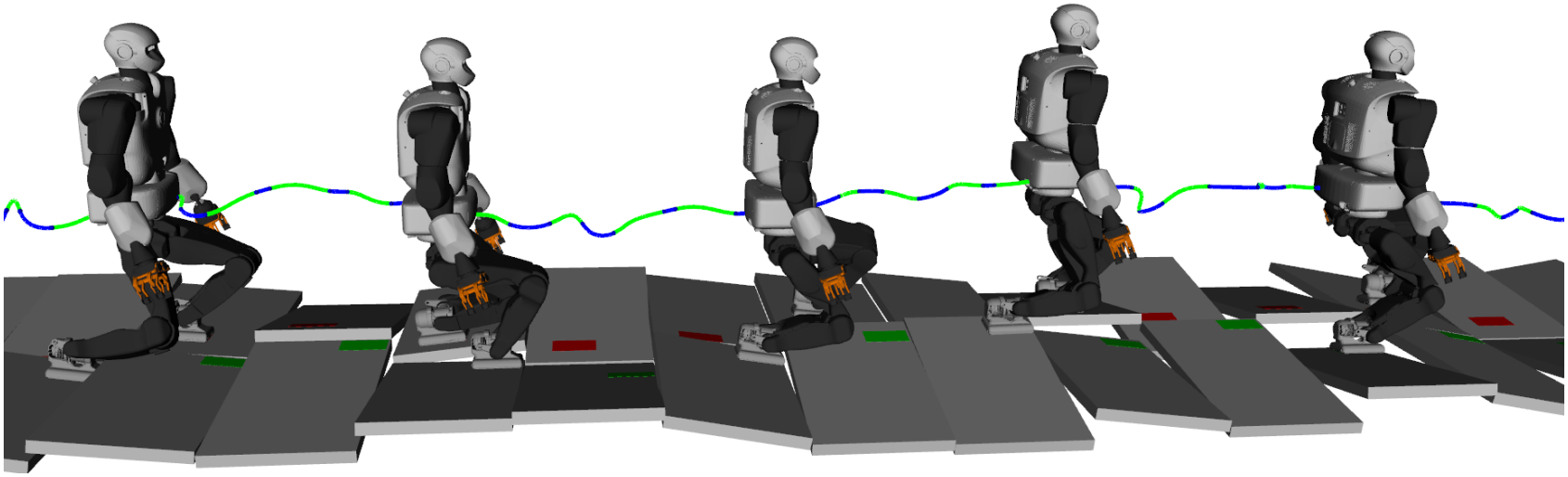
The abbreviation MF-RHP refers to our Multi-Fidelity RHP approach, while LG-RHP stands for our Locally-Guided RHP method. Furthermore, we use PH to refer to the Prediction Horizon.

Table 3: Average computation time of the baseline and the multi-fidelity RHP candidates with 1-step prediction horizon on the moderate slope terrain (CS1)

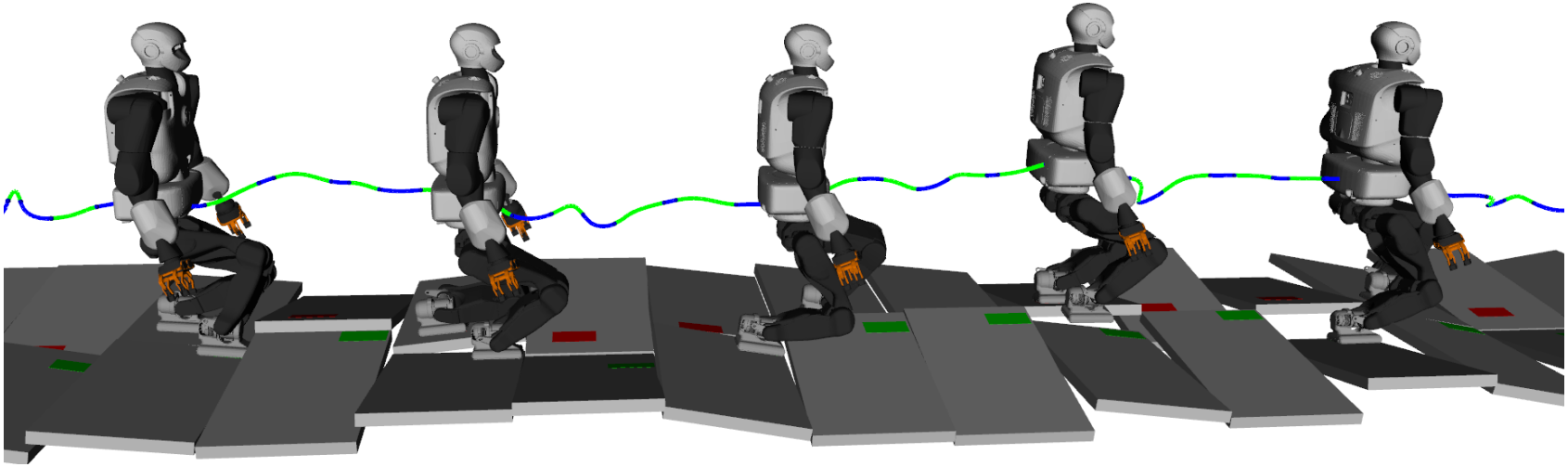
Method	Average Computation Time	Average Time Budget
Baseline	2.38 +/- 2.66s	1.77 +/- 0.33s
MF-RHP 2 (Rectangular foot)	1.03 +/- 1.06s	1.20 +/- 0.40s
MF-RHP 3 (Point foot)	**0.90 +/- 0.81s**	1.21 +/- 0.38s

Table 4: Average computation time of locally-guided RHP v.s. average time budget for the cycles that converged online.

Terrain	Average Computation Time	Average Time Budget
Moderate Slope (CS1)	**0.37 +/- 0.19s**	1.97 +/- 0.23s
Large Slope (CS2)	**0.36 +/- 0.23s**	2.01 +/- 0.48s



(a) Snapshots of the simulation for our second multi-fidelity RHP candidate with 1-step prediction horizon.



(b) Snapshots of the simulation for our third multi-fidelity RHP candidate with 1-step prediction horizon.

Figure 23: Snapshots of the simulation for our multi-fidelity RHP candidates with 1-step prediction horizon.

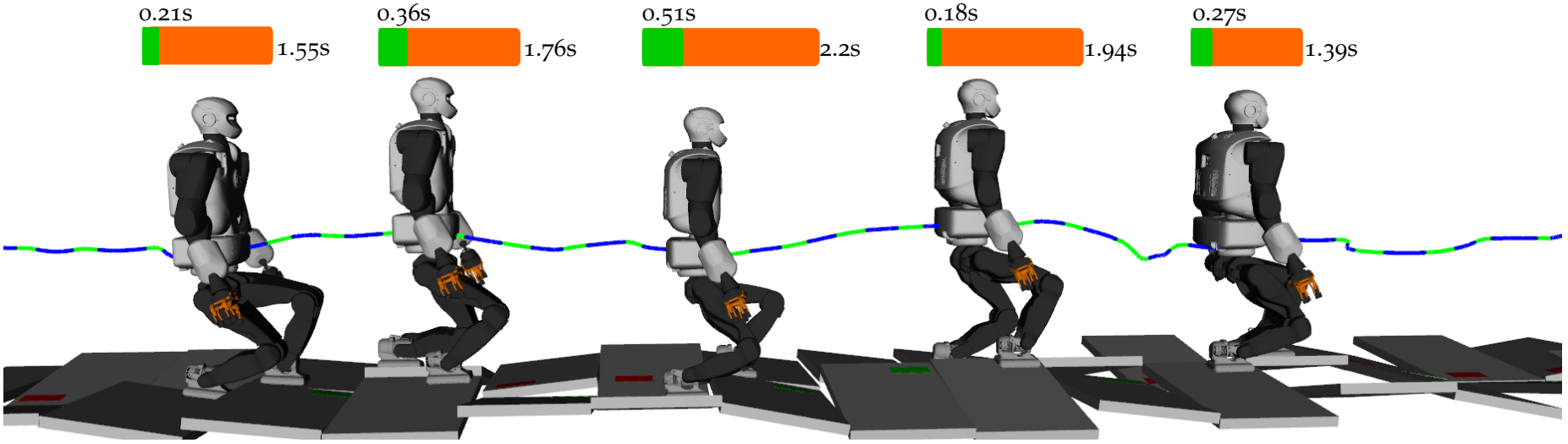


Figure 24: Snapshots of simulation result for locally-guided RHP on moderate slopes (5-12 degrees). We use the orange bar to represent the motion duration of the current cycle and the green bar as the computation time for the next cycle.

7.4 CASE STUDY 2 (CS2): LARGE SLOPE

In this section, we present the experiment result for the large slope terrain, where the robot cannot maintain static stability and has to traverse the terrain dynamically. We define each episode starts from the cycle when the large slope is captured inside the lookahead horizon and ending at the cycle when the robot gets off the large slope. For locally-guided RHP, we set the slack of the phase switching timing constraints (26) as $\epsilon = 0.15$, as empirically determined to give a good balance between the success rate and the computation speed. We list the computation performance of each RHP framework in Table 5.

7.4.1 *Computation Performance of the Baseline (Traditional RHP)*

In an offline setting (unlimited time budget), the baseline still can achieve a 100% episodic success rate on the considered large slope terrains. However, this requires the baseline to consider a 2-step prediction horizon, which can significantly increase the computation complexity of the TO problem. As a consequence, the baseline has 92.01% of the cycles fail to converge within the time budget.

7.4.2 *Computation Performance of Multi-Fidelity RHP*

On the other hand, we find that *multi-fidelity RHP candidates struggle to converge for the large slope terrain*. More specifically, similar to CS1, since our first multi-fidelity RHP ignores the angular dynamics in the prediction horizon, it can never complete a single episode on the large slope terrain. However, despite our second and third multi-fidelity RHP candidates consider convex relaxations of the angular dynamics, they still struggle to achieve a high convergence rate even in offline mode. For instance, when considering 1-step prediction horizon, both our second and third multi-fidelity RHP candidates fail to complete 59.87% to 63.49% episodes. Although we can slightly improve the episodic success rate by extending the length of the prediction horizon, our second and third multi-fidelity RHP candidates still fail to converge for almost half of the episodes (41.81% to 46.62%). This result suggests that the convex relaxations we employed in the prediction horizon may not be accurate enough to represent the momentum variation of the highly dynamics motion for traversing the large slope, and further investigation on the balance between the model accuracy and computation complexity is needed.

7.4.3 *Computation Performance of Locally-Guided RHP*

Despite the increased terrain complexity, our *locally-guided RHP still achieves the highest computation efficiency among all the RHP frameworks considered*. More specifically, our experiment result shows that our locally-guided RHP has 95.99% cycles successfully converged, and 95.0% of the cycles achieve online computation. Owing to the fast computation, our locally-guided RHP can also maintain online computation for 76.1% of the episodes. For the episodes that fail to achieve online computation consecutively, only 3.8% of them are due to time out, while the rest (20.1%) are caused by convergence failures. Furthermore, as indicated in Table 4, similar to CS1, our locally-guided RHP only consumes on average 18% of the time budget, demonstrating the potential of applying for real-world robot control.

In Fig. 25 and Fig. 26, we show the simulation results for our locally-guided RHP on the large slope terrain.

Table 5: Computation Performance for the Large Slope Terrain (CS2).

Method		Episodic Success Rate				Cycle-wise Success Rate			
		Success (Offline)	Success (Online)	Time Out	Fail to Converge	Success (Offline)	Success (Online)	Time Out	Fail to Converge
Baseline	1-Step PH	78.47%	0.25%	78.22%	21.53%	94.37%	22.93%	71.44%	5.63%
	2-Step PH	100.0%	0.0%	100.0%	0.0%	100.0%	7.99%	92.01%	0.0%
MF-RHP 1 (CoM)	1 to 3-Step PH	0.0%	-	-	-	-	-	-	-
MF-RHP 2 (Rectangle)	1-Step PH	40.13%	2.36%	37.77%	59.87%	83.0%	43.63%	39.37%	17.0%
	2-Step PH	52.66%	0.27%	52.39%	47.43%	89.90%	26.58%	63.32%	10.10%
	3-Step PH	53.38%	0.0%	53.38%	46.62%	91.81%	8.00%	83.80%	8.19%
MF-RHP 3 (Point)	1-Step PH	36.51%	4.89%	31.62%	63.49%	81.98%	52.50%	29.48%	18.02%
	2-Step PH	57.27%	0.77%	56.50%	42.73%	90.92%	37.54%	53.37%	9.08%
	3-Step PH	58.19%	0.0%	58.19%	41.81%	92.64%	20.65%	71.99%	7.36%
LG-RHP	-	79.9%	76.1%	3.8%	20.1%	95.99%	95.0%	0.99%	4.01%

The abbreviation MF-RHP refers to our Multi-Fidelity RHP approach, while LG-RHP stands for our Locally-Guided RHP method. Furthermore, we use PH to refer to the Prediction Horizon.

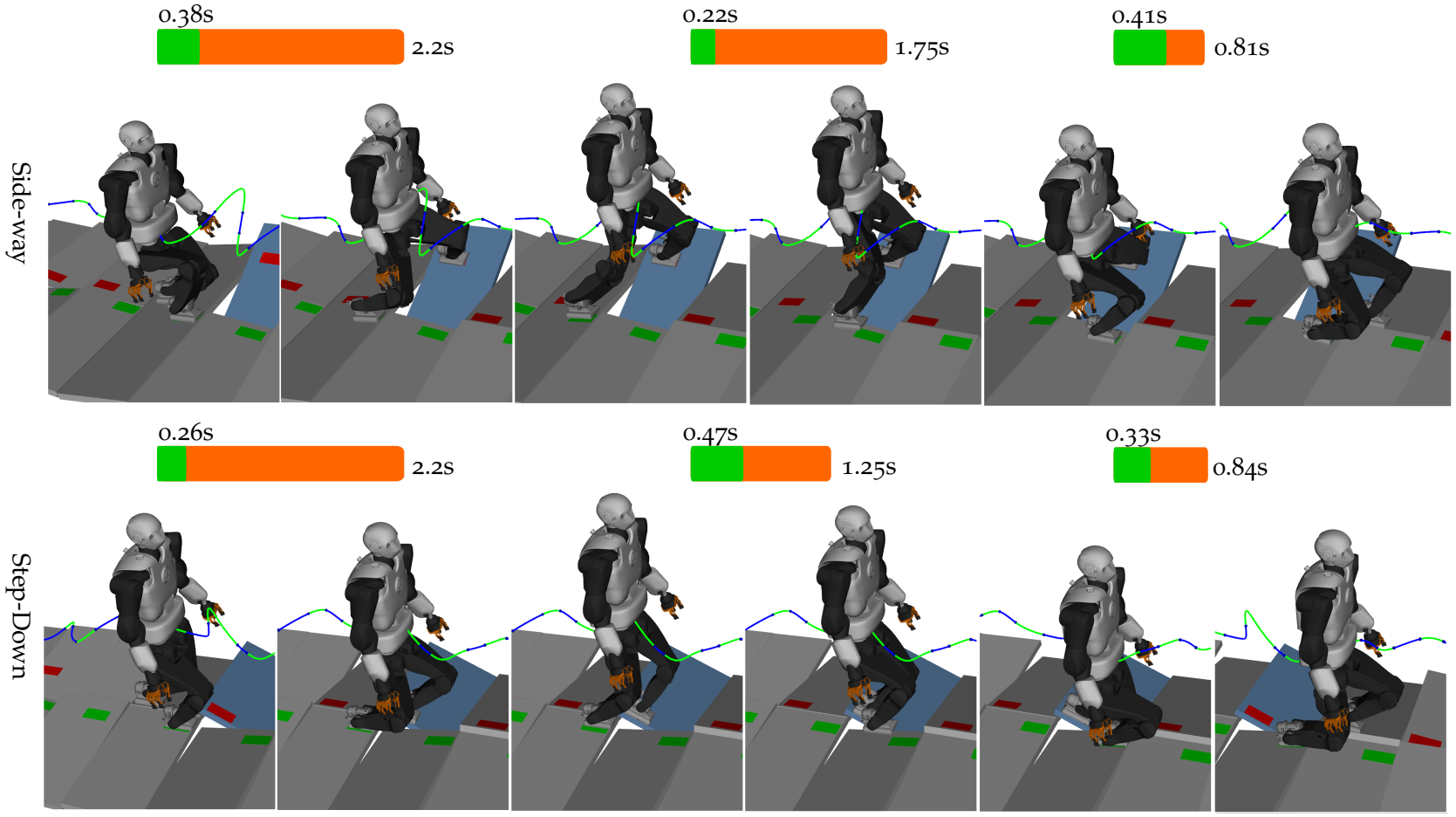


Figure 25: Simulation result for locally-guided RHP on the large slope terrain (side-way slope and step-down slope). The blue block is the large slope (25 degrees), while the rest are moderate slopes (5-12 degrees). The robot tends to build momentum to achieve dynamic balancing on the large slope. We show that locally-guided RHP can be used online, as the computation time of the next cycle (green bar) is smaller than the motion duration of the current cycle (orange bar).

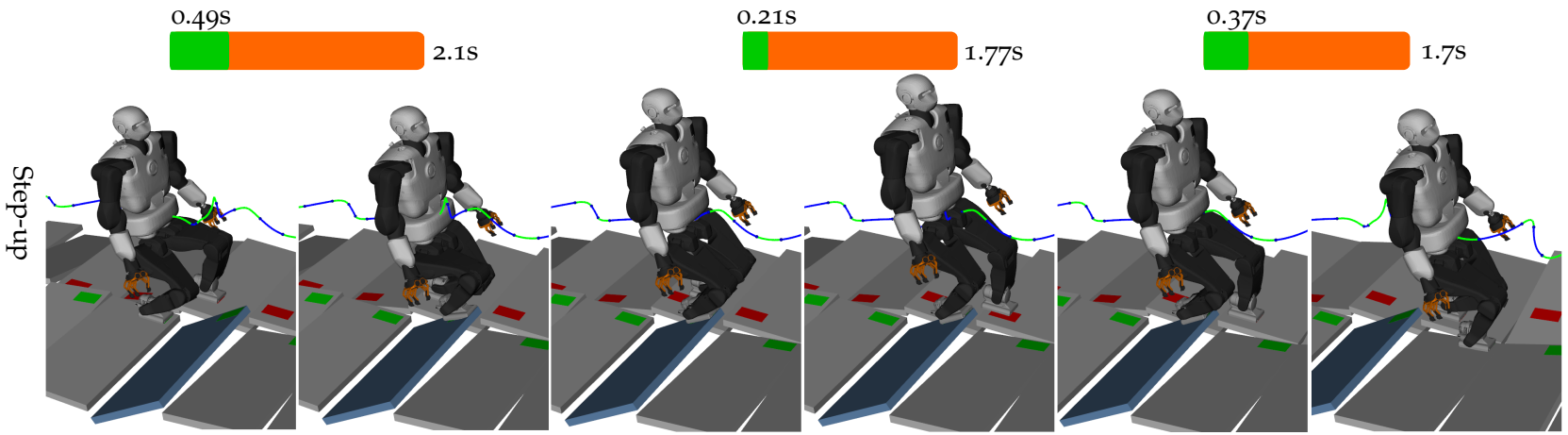


Figure 26: Simulation result for locally-guided RHP on the large slope terrain (step-up slope). The blue block is the large slope (25 degrees), while the rest are moderate slopes (5-12 degrees). The robot tends to build momentum to achieve dynamic balancing on the large slope. We show that locally-guided RHP can be used online, as the computation time of the next cycle (green bar) is smaller than the motion duration of the current cycle (orange bar).

7.4.4 *Improving Prediction Accuracy with Incremental Training Scheme*

In this section, we provide evidence to demonstrate our incremental training scheme described in Section 6.2.2 can improve the prediction accuracy of the oracle employed in locally-guided RHP.

We recall that we train separate oracles for the two case studies, namely the moderate slope terrain and the large slope terrain. The example terrain of these two case studies can be found in Fig. 22. In each training iteration, we test the episodic success rate of locally-guided RHP on the training environments of these two case studies, where the moderate slope has 3852 training environments, and the large slope has 6245 training environments.

In Table 6, we list the episodic success rate of locally-guided RHP achieved on the training environments with the oracle trained from different iterations of the data augmentation process. As we can observe from the table, in the first training iteration where we train the oracle with the initial dataset (no corrective data points are added), the locally-guided RHP only achieves 67.2% to 71.5% episodic success rate on the training environments. After the second and the third training iteration where we augment the dataset with corrective data points of interest, we significantly improve the episodic success rate of our locally-guided RHP to 80.3% and 79.5%. However, we also notice that, after the fourth and fifth training iteration, our incremental training scheme only marginally improves the episodic success rate to 82.1% and 81.0%.

Table 6: Episodic success rate for different iterations of the incremental training scheme described in Section 6.2.2 and Fig. 21-d. We consider two case studies, namely moderate slope and large slope (examples in Fig. 22), and we train separate oracles for these two case studies. In each training iteration, we test locally-guided RHP on the training environments of the two case studies, where we have 3852 training environments for the moderate slope case, and 6245 training environments for the large slope case.

Iteration Number	Moderate Slope (CS1)	Large Slope (CS2)
1	67.2%	71.5%
2	76.2%	75.3%
3	80.3%	79.5%
4	81.8%	80.4%
5	82.1%	81.0%

7.5 DISCUSSION

In this section, we discuss the advantages and disadvantages of multi-fidelity RHP and locally-guided RHP against the baseline (traditional RHP).

7.5.1 *Discussion of Baseline (Traditional RHP)*

From the result of the baseline, we first verify that considering an accurate system dynamic model in the prediction horizon can guarantee a high convergence rate (100% for the terrains we considered). This is expected as the accurate system dynamics model allows the prediction horizon to approximate the value function as accurately as possible. Furthermore, we also find that although the prediction horizon does not need to be infinitely long, having a prediction horizon with sufficient length is essential to the convergence of the baseline. For instance, our experiment result shows that the baseline only requires 1-step prediction horizon to achieve successful RHP on the moderate slope terrain. However, to traverse the large slope terrain where static stability cannot be maintained, the baseline may need 2-step prediction horizon. Despite the high convergence rate, the downside of the baseline is the long computation time due to the consideration of the non-convex centroidal dynamics model, which hinders its online usage.

7.5.2 *Discussion of Multi-Fidelity RHP*

To facilitate online multi-contact RHP, we explore the trade-off between the computation efficiency and the model accuracy in the prediction horizon. This gives rise to multi-fidelity RHP, where we reduce the TO complexity by employing a convex relaxed model in the prediction horizon. From our experiment results, we can draw the following conclusions. First, we find that the multi-fidelity RHP always fails to complete an episode if we only consider linear CoM dynamics in the prediction horizon (Candidate 1). This suggests that the convex relaxation employed in the prediction horizon cannot be arbitrary, and considering the angular dynamics is important. This finding leads to our second and third multi-fidelity RHP candidate, where we model the angular dynamics with a convex relaxation. Second, from the result of our second and third multi-fidelity RHP candidate, we can conclude that employing a valid convex relaxation in the prediction horizon can successfully improve the computation efficiency and increase the chances of achieving online computation. However, planning the prediction horizon with a relaxed model can inevitably affect the accuracy

of the value function approximated by the prediction horizon. This can introduce the risk of failing to converge, i.e., on the moderate slope, our second and the third multi-fidelity RHP fail to complete 27.5% and 20.52% episodes with 1-step prediction horizon. Although we can improve the convergence rate of our multi-fidelity RHP by extending the length of the prediction horizon, this increases the dimensionality of the TO problem, which in turn hinders online computation. Lastly, we realize that on the large slope terrain, our multi-fidelity RHP fails to complete about half of the episodes, and extending the length of the prediction horizon does not improve much on the convergence rate. This suggests that computing the prediction horizon with our proposed convex relaxations may lead to in-accurate value function approximations for the case of large slope terrain. We guess the inaccuracy comes from the following two factors. First, the proposed convex relaxations may not be tight enough to capture the momentum changes of highly dynamic motions [2]. Second, the manually fixed phase switching timings in the prediction horizon can be invalid for describing such dynamic motions. We believe further investigation regarding the trade-off between the computation efficiency and model accuracy is necessary.

7.5.3 *Discussion of Locally-Guided RHP*

To further improve the computation efficiency of multi-contact RHP, we propose locally-guided RHP, where we approximate the value function with a learned model. More specifically, we introduce an oracle to predict local objectives for achieving a given task, and we then construct local value functions to attract the execution horizon towards these predicted local objectives. This approach features a shortened planning horizon (only plans the execution horizon), and we demonstrate that locally-guided RHP can achieve the best online convergence rate (95% to 98.63% cycles converged online) compared to the baseline and the multi-fidelity RHP. Furthermore, we also show that locally-guided RHP only consumes on average 19% of the time budget (the motion duration of the previous cycle). This suggests that locally-guided RHP can be used for real-world robot control, as it leaves sufficient time for other overheads, e.g., data transmission.

However, locally-guided RHP still struggles in the following two cases. First, the oracle can have prediction errors due to imperfect fitting and insufficient data coverage. This can lead to inaccurate value functions, which direct the robot towards ill-posed states and cause convergence failures. Although we can mitigate this issue with an incremental training scheme that demonstrates recovery actions from unseen states, we find it hard to cover all the possible combinations of the robot state and environment models. To further improve the convergence rate of locally-guided

RHP, we can train the oracle with a Recurrent Neural Network (RNN) or impose safety constraints in the short-horizon TO. Second, although locally-guided RHP only computes execution horizon, it is still a nonlinear programming problem with no guarantee on computation time and can fail to converge online. To alleviate this issue, a viable option is to further reduce the number of decision variables by representing trajectories with parameterized curvatures, e.g., Bezier Curves [35].

Moreover, as mentioned in Section 7.2, we find that the data points for the two types of terrains exhibit different modalities. This can impose challenges when training a single Neural Network on the combined data set. Although we capture the two modalities by using separate Neural Networks, it is worthwhile to explore a more unified approach that can handle multi-modal data, e.g., using mixture density networks [9] or mixture of experts [116].

7.6 CONCLUSION

In this chapter, we have provided a simulation study to compare the computation performance of multi-fidelity RHP and locally-guided RHP against the baseline (traditional RHP). Based on the experiment result, we can draw the following conclusions. First, the result of the multi-fidelity RHP demonstrates that it is possible to achieve online computation by relaxing the model accuracy in the prediction horizon. However, this can downgrade the accuracy of the value function approximated by the prediction horizon, which may cause convergence failures. To improve the performance of multi-fidelity RHP, we believe future investigation on the balance between the computation efficiency and the model accuracy is important.

On the other hand, owing to the shortened planning horizon, our locally-guided RHP achieves the best computation efficiency among all the RHP frameworks. However, we find that the oracle can have prediction errors and lead to convergence failures. To alleviate this issue, we employ an incremental training scheme to add data points from the states that cause convergence failures. We still find it hard to achieve 100% prediction accuracy with this approach, showing that further investigation on improving the learning accuracy is necessary.

In our methods, we assume that the combinatorial aspect of the locomotion problem—the gait pattern (order of contact activation) and the sequence of the contact surfaces in which the robot will step upon—are given. Although our RHP methods can successfully generate feasible locomotion plans on the scenarios we considered, we believe taking into account the combinatorial aspect of the locomotion problem is important for planning motion for more complex cases, i.e., the terrains or tasks which require the robot to traverse with acyclic gait patterns [81, 109] or the robot has

to decide a feasible sequence of contact surfaces that the robot will step upon among a couple of potential options [26, 87]. In Chapter 9, we will present one step towards addressing the combinatorial aspect of the locomotion problem, where we focus on investigating the gait pattern selection issue of legged robots. In the next chapter, we will focus on presenting our real-world robot experiments to demonstrate the efficacy of the locally-guided RHP in achieving online receding horizon planning of contact and motions plans.

8

REAL-WORLD DEMONSTRATION OF ONLINE MULTI-CONTACT RECEDING HORIZON PLANNING

In the previous chapter, we have presented a simulation study that evaluates the computation performance of our online RHP approaches for computing contact and motion plans. Based on our simulation results, we find that locally-guided RHP features the best computation efficiency among all the RHP frameworks considered (see our experiment results in Table 2 and Table 5). This computation advantage suggests the potential of using locally-guided RHP to achieve online receding horizon planning of contact and motion plans in the real-world scenarios where online motion adaption is critical, i.e., traversing uneven terrain with unexpected changes. In this chapter, we carry out real-world robot experiments to demonstrate the efficacy of locally-guided RHP in achieving online receding horizon planning of contact and motion plans. These experiments are achieved on our torque-controlled humanoid robot platform Talos [90]. In Section 8.1, we describe the details of the software implementation. Then, in Section 8.2, we present the result of our real-world experiments. Furthermore, we find that the limitations of the robot state estimator and the controller can introduce challenges to achieve our robot experiments. In Section 8.3, we discuss these challenges, as well as our engineering solutions and future directions to address them.

8.1 SOFTWARE IMPLEMENTATION

To achieve our robot experiment, we build a software framework composed of the following modules (see Fig. 27): 1) a planning node that computes the motion plan in an online receding horizon fashion using locally-guided RHP based on the perceived environment, 2) a plan interpreter that coordinates the execution of the motion plan, and 3) a robot control stack that computes control commands to track planned motions.

The interplay between these modules is described as follows. At the beginning of each cycle ($t = 0$), the plan interpreter will start executing the motion already planned for the current cycle, while in the meantime requesting the planning node to compute the motion plan for the next cycle. For the planning side, once the planning node receives the request from the plan interpreter, it will firstly query the terrain perception module to provide the terrain model that is currently ahead of the robot. In our work, we realize the terrain perception based on the VICON motion capture system, where

we detect the state of a VICON marker plate to indicate different terrain geometries. We build our VICON system with 32 VICON Vero cameras to cover our experiment area (12 meters $\times$ 5 meters). Given the perceived terrain model, the planning node will use locally-guided RHP to predict the local objective and then plan the motion for the next cycle. Similar to the setup of our simulation studies (Section 7.1), we assume the robot can always track the planned trajectories without having large deviations. Hence, we define the motion plan for the next cycle always starts from the terminal state of the current cycle. To achieve online receding horizon planning, it is expected that the planning node can compute the motion plan for the next cycle within the duration T of the motion being executed for the current cycle. Once the planning node finishes the computation of the motion plan for the next cycle, it will send the computed motion plan to the plan interpreter, and the plan interpreter will store the motion plan for the next cycle in a buffer to wait for execution.

Apart from requesting the motion plan for the next cycle, another task of the plan interpreter is to execute the motion planned for the current cycle. This is achieved by constantly streaming a desired robot state for the robot to reach in each control loop of the robot at a rate of 500Hz. In our case, the desired robot state includes a reference CoM state and a reference swing foot state. The reference CoM state is achieved by interpolating the planned centroidal trajectory based on the time step t of the current control loop. To decide the reference swing foot state, we firstly establish a swing foot trajectory between the planned contact locations, and then we obtain the reference swing foot state by querying the swing foot trajectory according to the time step t of the current control loop. Given the desired robot state, the robot control stack will then use a whole-body inverse dynamics controller to compute the desired torque command for each joint based on the desired robot state from the plan interpreter and the state feedback from the state estimator. To implement these torque commands, each joint has a local torque controller that can track the desired torque commands based on the torque sensor feedback from each joint at a rate of 1000Hz. Once the plan interpreter finishes the motion execution of the current cycle, it will load the motion plan for the next cycle from the buffer and start the motion execution of the new cycle (reset the internal timer to $t = 0$). This will also trigger the plan interpreter to request the planning node to compute a new motion plan.

Our software implementation is based on the ROS framework [89], and the communication between the planning node and the plan interpreter is achieved through the ROS subscriber/publisher protocol. Furthermore, we implement the planning node in Python as described in Chapter 7. The plan interpreter is a customized software developed by ourselves and written in C++. The whole-body controller, the state estimator, and the local joint torque controller are provided by PAL Robotics.

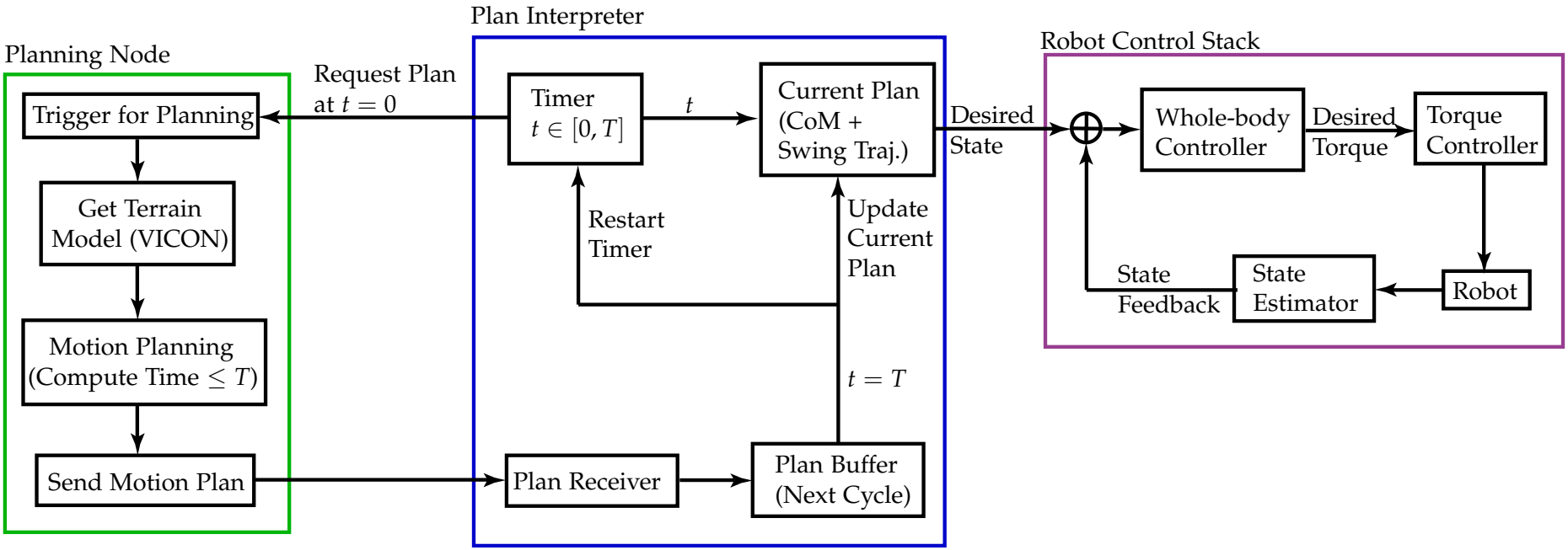


Figure 27: Software implementation for our real-world robot experiments on Talos. At the beginning of each cycle, the plan interpreter will request the planning node to compute the motion plan for the next cycle, while start executing the motion already planned for the current cycle. Once the planning node receives the request from the plan interpreter, the planning node will use locally-guided RHP to plan the motion for the next cycle. To achieve online receding horizon planning, it is expected that the planning node can compute the motion plan for the next cycle before the robot finishes the motion execution of the current cycle. To execute the motion plan for the current cycle, we design the plan interpreter to constantly stream a desired robot state for the robot to reach in each control loop of the robot at a rate of 500Hz. Given this desired robot state, the whole-body controller involved in the robot control stack will then computes the desired torque command of each joint based on the desired robot state and the state feedback from the state estimator. To achieve the desired torque command, each joint has a local torque controller to track the torque commands based on the torque sensing from each joint. This local joint torque controller runs at a rate of 1000Hz.

In this section, we present the results of our real-world robot experiments. To highlight the benefit of achieving online receding horizon planning, we consider the scenarios where the environment can change during run-time and the robot has to adapt its motion on-the-fly to achieve reliable and continuous operation. The video of these experiments can be found at https://youtu.be/oMo_50XIE24. Same as our simulation studies (Chapter 7), we implement locally-guided RHP in Python, and we use the interior-point method of KNITRO 10.10 [12] to solve optimization problems, along with the automatic differentiation framework CasADi [3] to provide the gradients and the Hessians to the optimization solver. The computations are carried out on a desktop with an Intel i9 CPU (3.6GHz) and 64GB memory.

Specifically, as shown in Fig. 28 and Fig. 29, we firstly consider a scenario where we change the flat surfaces to an up-and-down hill terrain along the pathway of the robot. During the first few cycles, the preview of the environment is considered as a flat surface (covered by a curtain). While the robot is moving forward, the flat region changes to an up-and-down hill terrain by removing the curtain. The planning node notices the change of the terrain by detecting the VICON marker plate, and updates the environment model accordingly. During this experiment, locally-guided RHP successfully achieved online computation of the contact and motion plans that are consistent with the latest terrain condition perceived by the robot. For instance, the average computation time of locally-guided RHP is 0.22 +/- 0.076 seconds, which is much smaller than the time budget (3.5 seconds). This fast computation speed allows the robot to safely traverse this changing environment. In Fig. 28 and Fig. 29, we show the snapshots of this experiment, as well as the motion plan generated in each cycle along with the terrain model perceived in that cycle.

In Fig. 30, we demonstrate another changing environment scenario, where we add a stair during the robot operation. Same as in the previous scenario, the locally-guided RHP also achieved online receding horizon planning in this scenario. The average computation time is 0.23 +/- 0.1 seconds, while the time budget is 3.5 seconds. This enables the robot to successfully overcome the newly introduced stair.

Furthermore, we also perform real-world experiments on challenging uneven terrains, such as continuously walking on 1) random slopes where the blocks are oriented around either the y-axis or the diagonal axis, 2) up-and-down hill terrain, and 3) the v-shape terrain. The snapshots of these experiments can be found in Fig. 31 and Fig. 32. The inclination of these slopes is 10 degrees. In these experiments, our locally-guided RHP successfully achieves online computation for all the cycles, where the average computation time is 0.21 +/- 0.06 seconds and the time budget is 3.5 seconds.

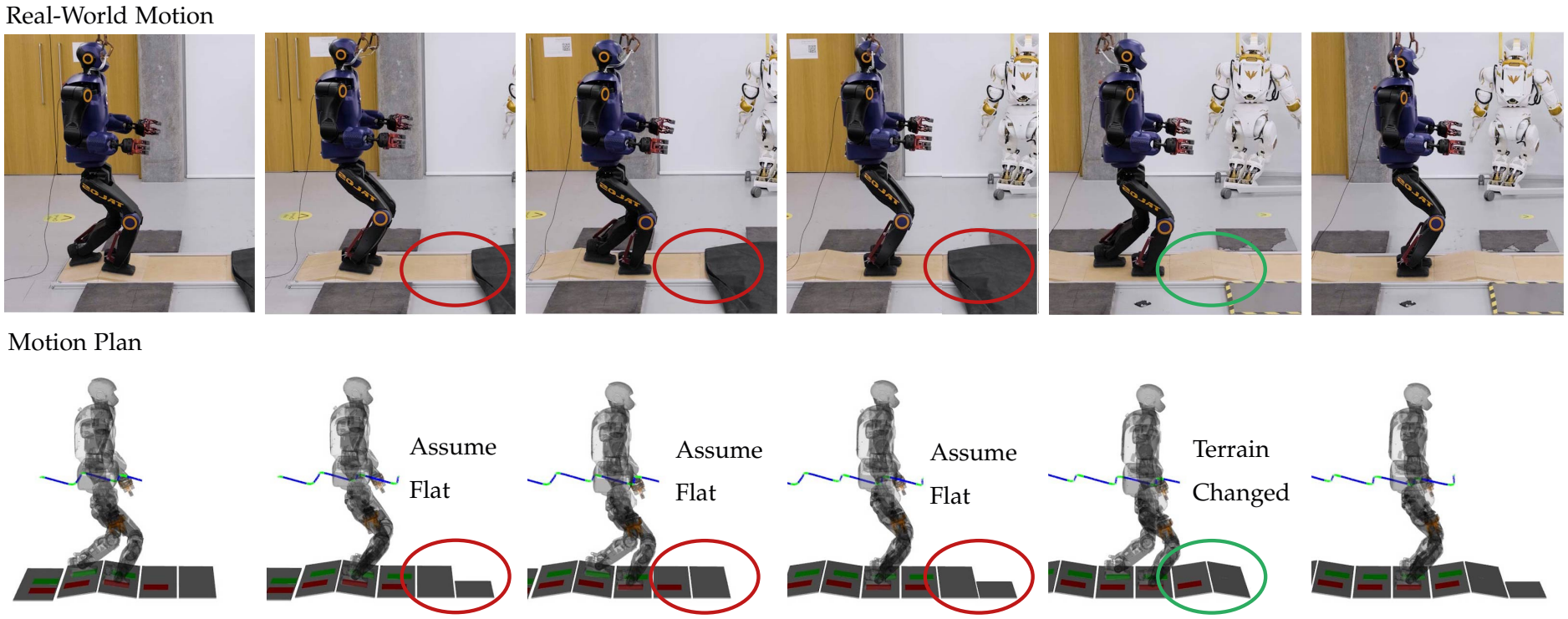


Figure 28: Snapshots for our first real-world experiment in changing environments and the motion planned in each cycle (part A). In this scenario, we change part of the environment from flat surfaces to an up-and-down hill terrain during run-time. We indicate this terrain change by placing a VICON marker next to the terrain. Once the planning node detects the VICON marker plate, it will modify the terrain model accordingly. Owing to the fast computation, our locally-guided RHP successfully achieved online receding horizon planning in this changing environment, where the average computation time of locally-guided RHP is 0.22 +/- 0.076 seconds and the average duration of the motion being executed in each cycle (time budget) is 3.5 seconds. This allows the robot to reliably traverse the terrain. The robot moves from left to right. The inclination of each slope is 10 degrees. This figure shows the first part of the full motion sequence and the rest of the motion sequence is shown in Fig. 29.

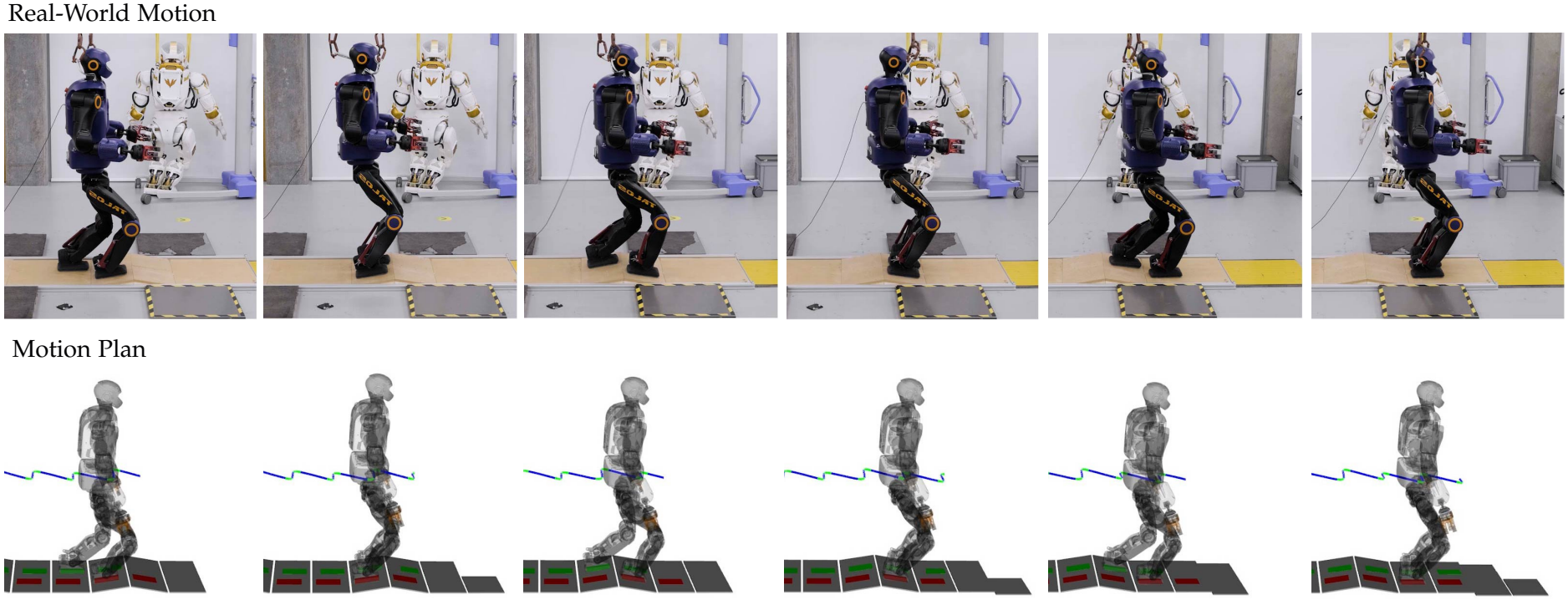


Figure 29: Snapshots for our first real-world experiment in changing environments and the motion planned in each cycle (part B). This figure shows the motion sequence of our first real-world experiment in changing environments continued from Fig. 28. Although the robot encounters an unexpected change of the environment (changing the flat surface to an up-and-down hill), the fast computation of locally-guided RHP enables the robot to online adapt its motion to the changed environment and safely traverse the terrain. In this experiment, the average computation time of locally-guided RHP is 0.22 +/- 0.076 seconds, and the average duration of the motion being executed in each cycle (time budget) is 3.5 seconds. The robot moves from left to right.

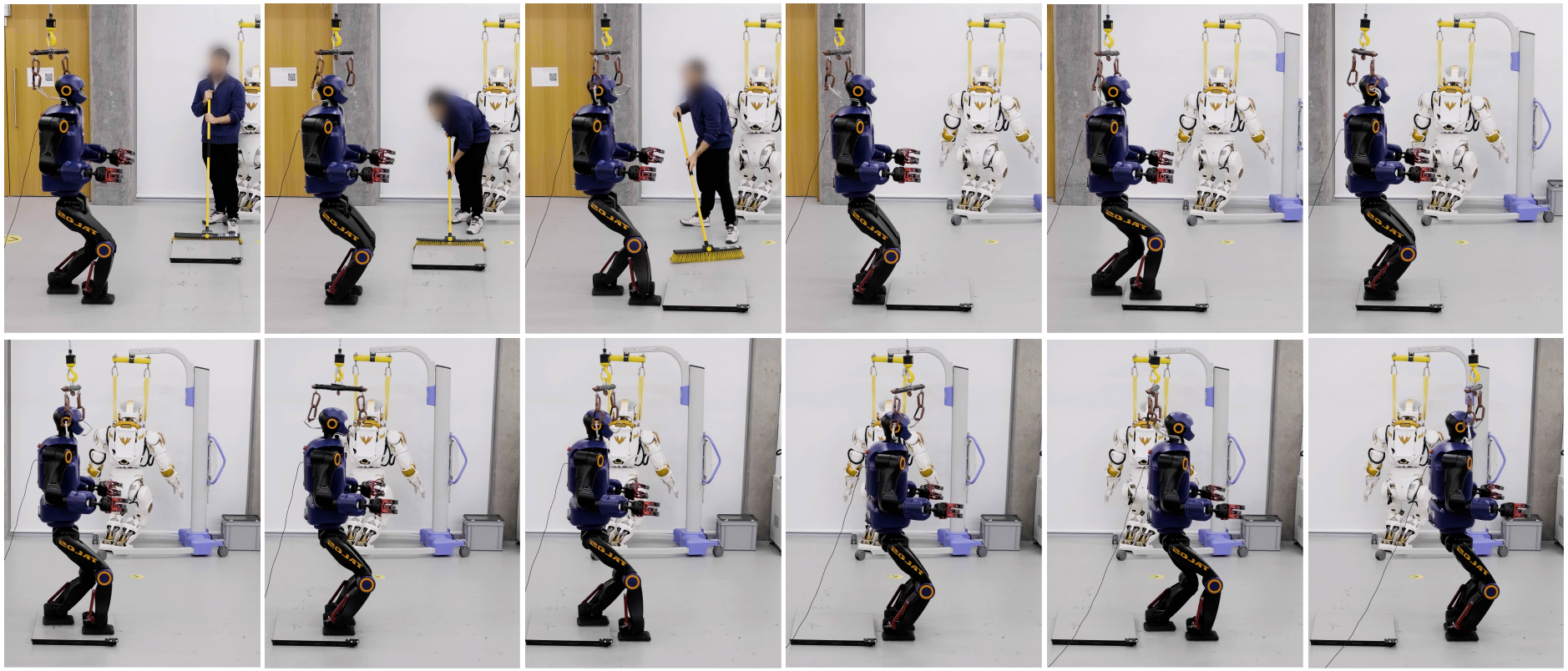


Figure 30: Snapshots of our second real-world experiment in changing environment. In this case, we add a stair (4cm height) while the robot is walking. The planning node detects the stair based on the position measurement of the VICON marker plate attached to the stair. In this experiment, our locally-guided RHP successfully achieved online computation in each cycle, which allows the robot to safely overcome the stair. The average computation time of locally-guided RHP is 0.23 +/- 0.1 seconds, while the average duration of the motion being executed in each cycle (time budget) is 3.5 seconds. The robot moves from left to right, top to down.

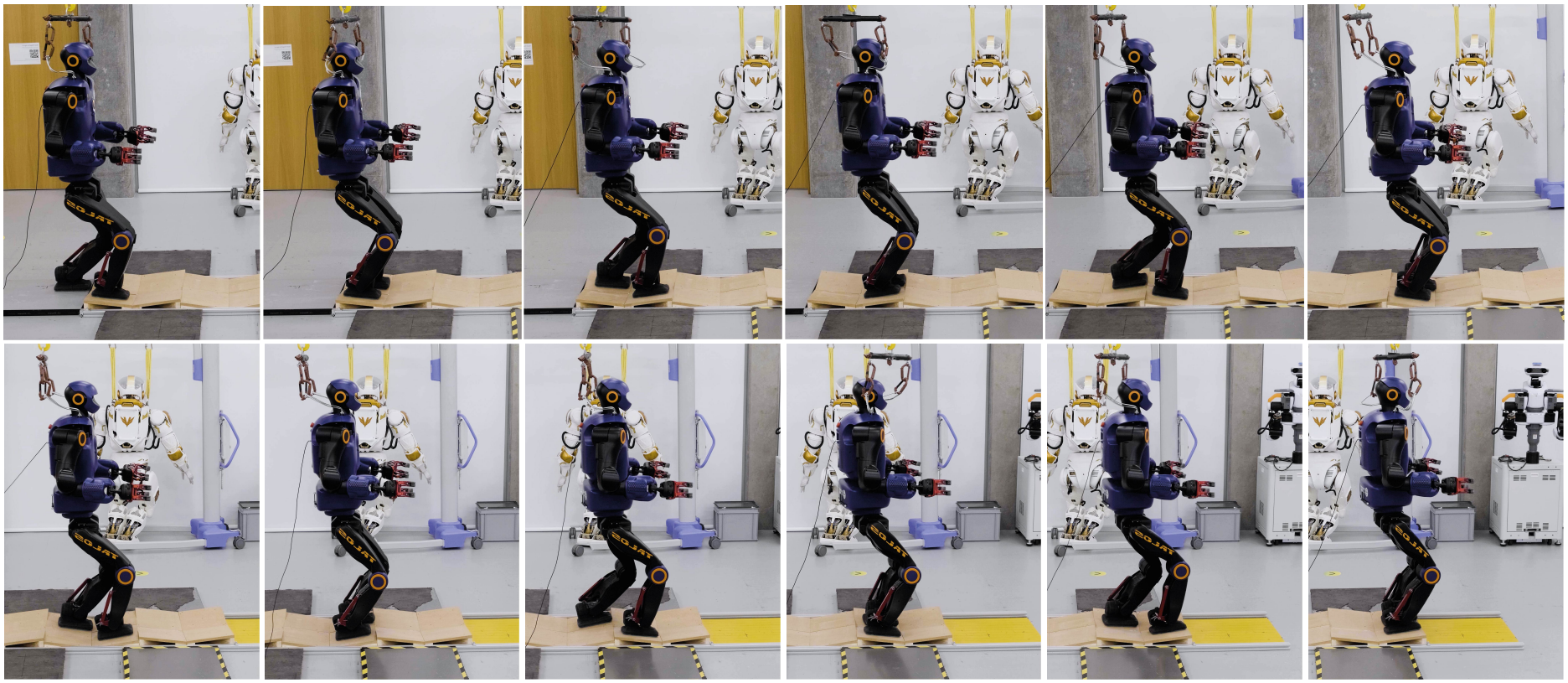


Figure 31: Snapshots of our real-world experiment on random slopes. The slopes are oriented around either the y-axis or the diagonal axis. The inclination of each slope is 10 degrees. In our challenging uneven-terrain experiments, the average computation time of locally-guided RHP is 0.21 +/- 0.06 seconds, while the average duration of the motion being executed in each cycle (time budget) is 3.5 seconds. The robot moves from left to right, top to down.

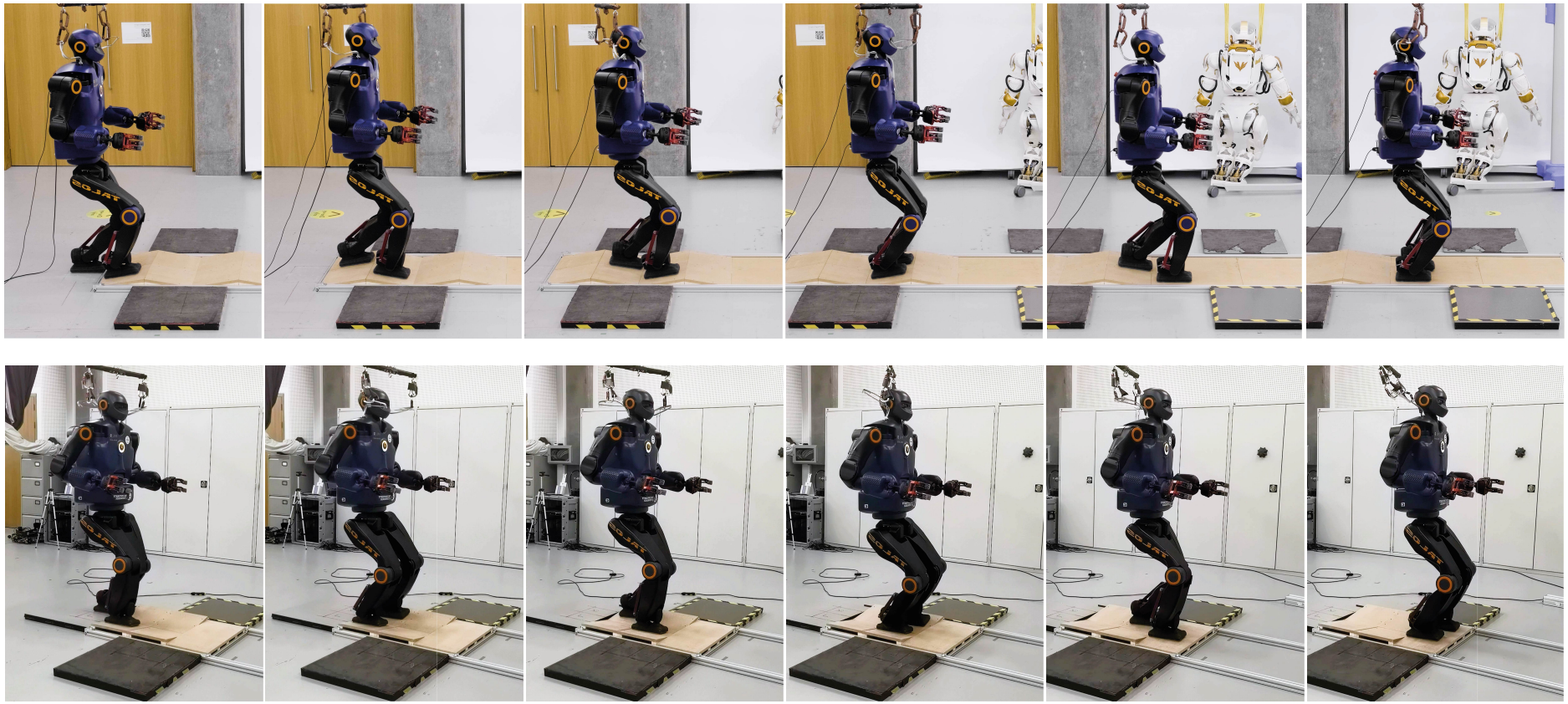


Figure 32: Snapshots of our real-world experiments on up-and-down hill terrain (top) and the v-shape terrain (down). The inclination of each slope is 10 degrees. In our challenging uneven-terrain experiments, the average computation time of locally-guided RHP is 0.21 +/- 0.06 seconds, while the average duration of the motion being executed in each cycle (time budget) is 3.5 seconds. The robot moves from left to right.

8.3 CHALLENGES IN REAL-WORLD EXPERIMENTS

To successfully execute the motion plan on the real robot, it is important to have a state estimator that can provide accurate state feedback on the robot, and a whole-body controller that can track the planned motion precisely. Nevertheless, during our real-world tests, we realize that the state estimator can have drifts and the whole-body controller can have tracking errors. Given these issues, the robot can have the risk to fall during our real-world tests. In this section, we will discuss these real-world challenges and present our attempts and future directions to address them. We believe further investigations on improving the performance of the state estimator and the whole-body controller are also important for achieving reliable legged systems in the real world.

8.3.1 *State Estimation Drifts*

During motion execution, the state estimator provides state feedback in a fixed reference frame called odometry frame, and the whole-body controller also tracks the motion plan in this odometry frame. In our case, the odometry frame is placed in between the two feet of the robot, and it is parallel to the ground where the robot feet stay upon.

To execute the motion plan, we firstly need to transform the motion plan from the planning frame to the odometry frame. For simplicity, we set the planning frame always overlapping with the odometry frame (the transformation is an identity matrix).

As we can imagine, in order to achieve successful real-world operation, it is important to ensure that the transformation between the planning frame and the odometry frame is always a constant, and the state estimator can also provide accurate feedback in this odometry frame (see an example in Fig. 33-a).

Nevertheless, during our experiments, we realize that the transformation between the planning frame and the odometry frame is often non-static. For instance, we observe that the odometry frame defined by the state estimator can gradually drift down along the z-axis after the robot makes a new contact. The reason of having this drift is mainly due to the inaccurate contact detection of the robot. More specifically, when having inaccurate contact detection, the state estimator will struggle to accurately capture the swing foot height when it tries to make contact with the ground. Such error in the foot height will then affect the state estimator to accurately estimate the actual ground level based on the height of the foot staying in contact, and thus results in drifts of the odometry frame.

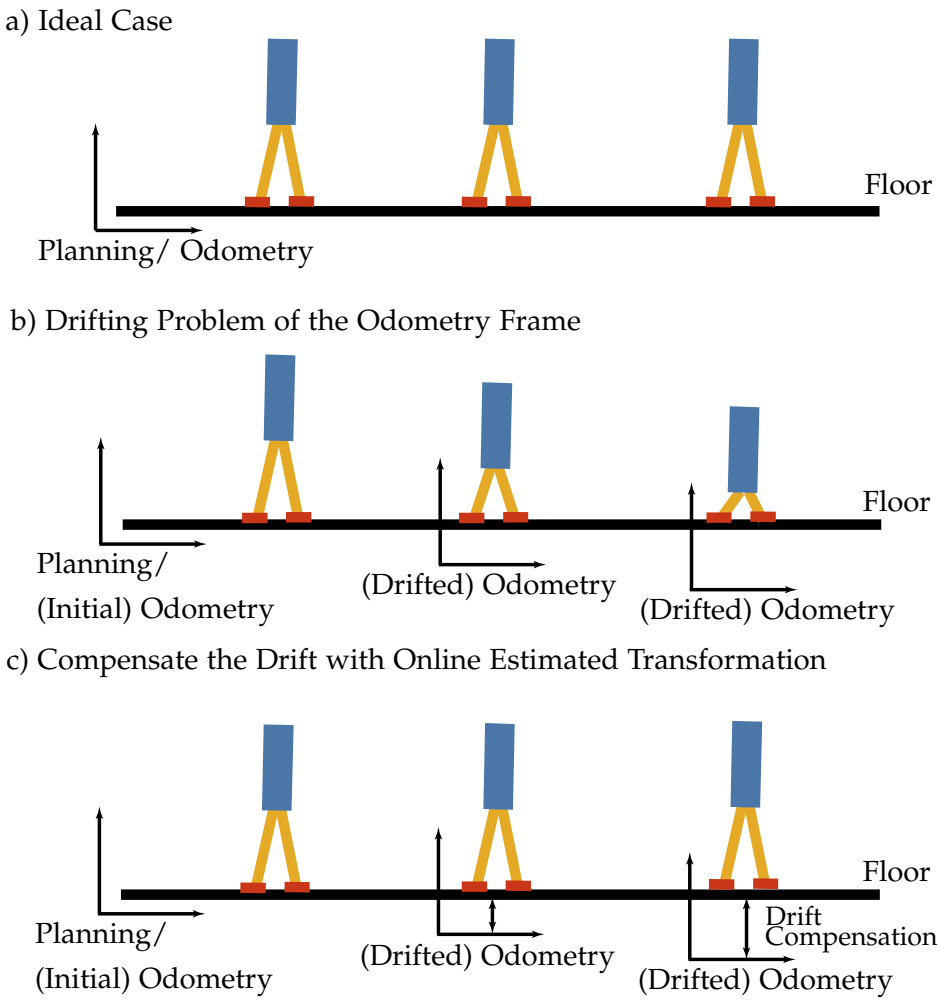


Figure 33: a) Before executing the motion plan, we need to transform the motion plan from the planning frame to the odometry frame (defined by the state estimator). During motion execution, the transformation between these two frames should be always static, i.e., assuming they are the same. b) In reality, the odometry frame can gradually drift down. This will cause the robot keeps lowering its CoM, as the motion plan is also drifting down with the odometry frame. With the robot keeps lowering its body, the robot may hit joint limits. c) To compensate for the drift, we can online estimate the transformation between the planning frame and the odometry frame, and then correct the drift by applying this transformation to the motion plan that will be sent to the controller.

Due to the drift of the odometry frame, the actual motion plan we execute will also flow with the odometry frame to drift down (viewing from the planning frame). This is because we assume the odometry frame is always the same as the planning frame, and there is no compensation to cope with the drift when we transform the motion plan from the planning frame to the odometry frame. As a consequence, in the real world, we can observe that the robot will continue to lower its CoM position as the target CoM position is drifting down with the odometry frame (see an example in

Fig. 33-b). As the robot keeps lowering its body, the joints of the robot legs can hit their kinematics limits and cause failures. Similarly, the target footstep locations are also drifting down with the odometry frame (penetrating into the ground). However, given the fact that the floor is always static and solid, the drifted target footstep locations can never be reached. Instead, the robot feet exhibit early contact events. These early contact events can generate impacts that may damage the hardware, and we definitely prefer to avoid these damages.

To address the drifting problem of the odometry frame, a viable approach is to constantly estimate the transformation between the planning frame and the drifted odometry frame. Then, in every control loop, we can compensate for the drift of the odometry frame by applying this transformation to the desired motion plan (see an example in Fig. 33-c).

To estimate the transformation between the planning frame and the drifted odometry frame, we firstly attempt an engineering solution. In this engineering solution, we assume that in the real world, when the robot tries to make a step, it can always place its foot at the planned contact location. Given this assumption, we can compute the amount of the z-axis drift of the odometry frame by comparing the estimated foot height in contact (in the odometry frame) against the height of the planned contact location (in the planning frame). Then, based on this computed z-axis drift of the odometry frame, we can construct the transformation between the planning frame and the drift odometry frame, and apply this transformation to the desired motion plan to compensate for the drift. This engineering solution has helped us to achieve all the robot experiments in Section 8.2.

Nevertheless, when walking on uneven terrain, due to imperfect control and foot slippage, the robot may place its foot in a slightly different location than the planned location. As a result, the actual height of the foot in contact can become different from the height of the planned contact location, and thus the z-axis drift of the odometry frame estimated by our engineering solution may become inaccurate and introduce instability to the robot.

To obtain a more accurate estimation of the transformation between the planning frame and odometry frame, we also attempt a more systematical approach, in which we use a ROS software called robot localization package [69] to estimate the transformation. To give more detail, in order to compute the transformation, the robot localization package requires the inputs of the robot states in both the planning frame and the odometry frame. These robot states can be the 6D pose of a robot link. In our case, we feed the robot base (pelvis) states into the robot localization package. Then, the robot localization package will continuously fuse these robot states through

an extended Kalman filter [44] and update the transformation between the planning frame and the odometry frame.

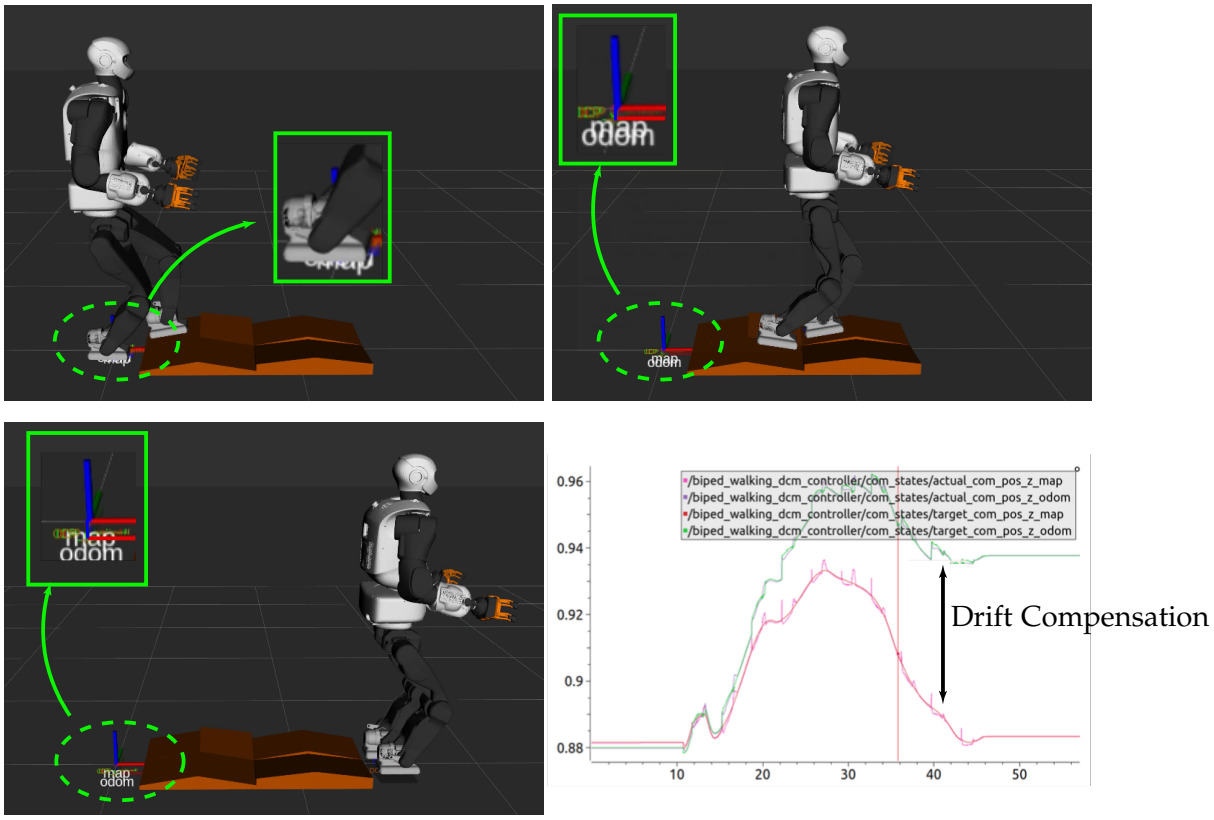


Figure 34: Simulation result for correcting the drift of the odometry frame using the robot localization package. Before making any contacts, the planning frame and the odometry frame are consistent (they overlap in the top-left image). As the robot starts creating new contacts, the odometry frame will start drifting down (the two frames get separated in the top-right and bottom-left images). We recall that, if there is no compensation to cancel this drift, the robot will continuously lower its CoM, and then reach its kinematics limit and fail. In our case, we constantly compensate for the drift by estimating the transformation between the planning frame and the odometry frame, and then apply this transformation to the motion plan being executed. For instance, as shown in the bottom-right image, the estimated transformation can compensate the drift by lifting the motion plan in the odometry frame (the green curve), and this can allow the robot to successfully achieve the desired CoM as we specified in planning frame (the pink curve).

To verify the effectiveness of this approach, we firstly carry out simulation studies with a physical simulator provided by PAL Robotics. In this simulation study, the robot base state in the odometry frame is estimated by the robot state estimator, while the ground-truth state provided by the simulator is regarded as the robot base state in the planning frame (the world frame). As Fig. 34 illustrates, although the odometry frame keeps drifting down, our method can successfully capture the transformation

between the planning frame and the odometry frame, and then constantly compensate for the drift to the motion plan sending to the robot for execution.

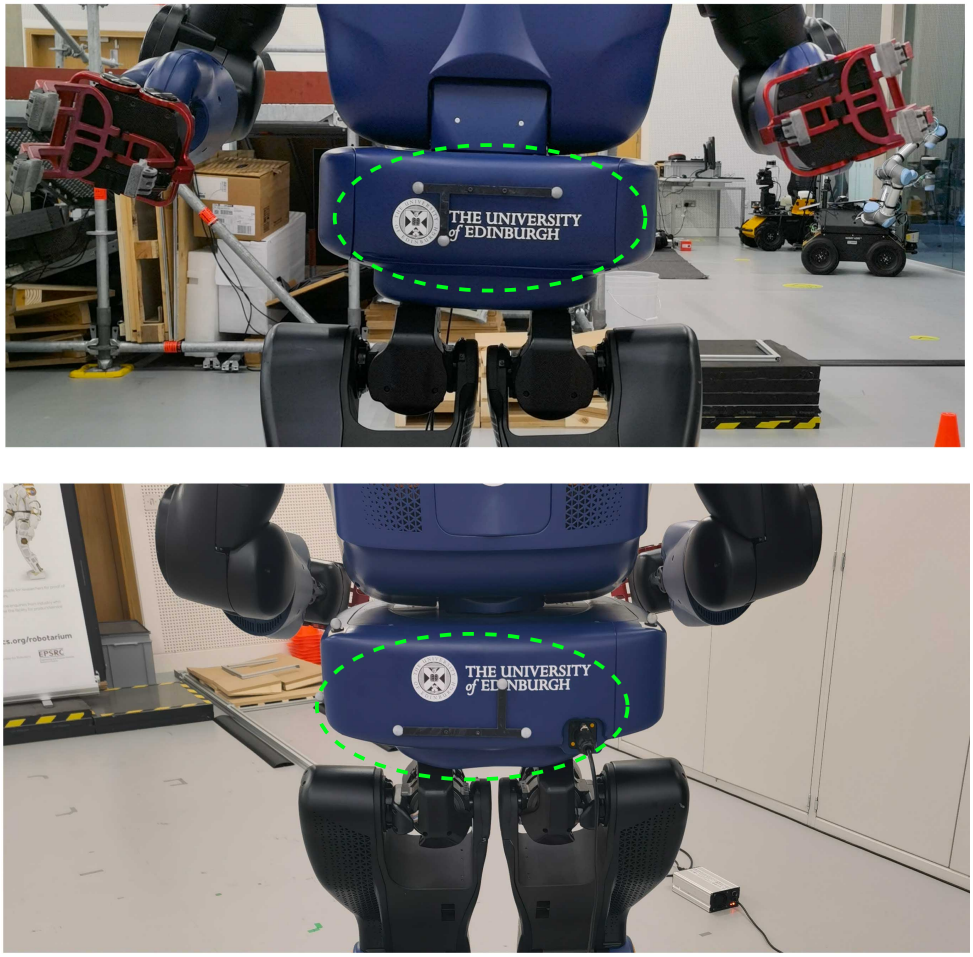


Figure 35: Placement of VICON markers. Top image: markers mounted on the front side of the robot base. Bottom image: markers mounted on the back side of the robot base. The markers are supported by 3D-printed frames (black).

Given the success of our simulation verification, we then start applying this approach in the real world. In the real-world case, we still use the robot state estimator to provide the robot base state in the odometry frame. For getting the robot base state in the planning frame (the world frame), we use an VICON motion capture system. To give more detail, in order to allow VICON to measure the robot base state we firstly need to place a set of reflective markers on the robot base (see our setup in Fig. 35), and then provide the local coordinates of these markers in the local robot base frame (the frame which defines the 6D pose of the robot base). Then, by comparing these local

coordinates with the global marker positions captured by the infrared cameras, the VICON system can estimate the robot base state in the measurement frame (in our case, the planning frame). These computations are achieved by the proprietary VICON software.

Nevertheless, when we test our method in the real-world, we find that the transformation between the planning frame and the odometry frame are often not perfectly aligned, i.e., the orientation (roll, pitch, yaw) of these two frames can have 1-2 degrees mismatch. Due to this mismatch, applying the transformation will result in a titled motion plan in the odometry frame, and executing this titled motion plan will cause the robot to fall. For example, Fig. 36 illustrates the effect of having an orientation mismatch around the positive roll axis. As we can observe, this mismatch results in the robot tilting its CoM towards the right foot of the robot. It is worth to note that, although the orientation mismatch can be just around 1 degree, due to the long leg of the robot (about 0.88m), the CoM can have about 1.5 to 2 centimeter shift on the horizontal plane. As we can imagine, during walking, the robot can easily loose the balance when the right foot breaks contact with the ground, as the tilt brings the CoM away from the support region of the left foot.

The mis-alignment between the planning frame and the odometry frame can come from multiple sources. First, we speculate that the robot base state measured by the VICON may have errors. These errors can be the result of many reasons. For instance, we may have in-accurate local coordinates of the VICON markers defined in the robot base frame. Although we try to define these local coordinates as accurately as possible (for example, calculating them based on the CAD model of the robot base), we may still have errors due to the imperfect manufacturing of the robot base and the 3D-printed framework used for supporting the markers. Additionally, the global marker position measured by the infrared cameras of VICON can also have errors or jumps (due to occlusion). These issues can together affect the accuracy of the robot base state in the planning frame (measured by VICON). On the other hand, the robot also has an Inertial Measurement Unit (IMU) produced by Orientus. This IMU is used to measure the robot base orientation in the odometry frame, and it can also have measurement errors. To give more detail, in Talos, the IMU is not directly mounted in the robot base, but placed in the robot torso. When estimating the robot base orientation in the odometry frame, the robot firstly gets the IMU reading of the robot torso, and then compute the robot base orientation based on the IMU reading and the position reading of the joint that connects the robot torso and the robot base. This calculation involves a kinematics chain that may bring errors. Furthermore, the IMU is mounted in the robot torso with a 3D-printed support. This support may not perfectly align with the robot torso frame (the frame defines the robot torso orientation). To improve the accuracy of

these measurements, we suggest to perform a calibration in the future. For instance, we can calibrate the robot base orientation measured in both the planning frame and the odometry frame with respect to a ground-truth orientation, i.e., constraining the robot base in a known posture, and calibrate the readings with respect to this known posture.

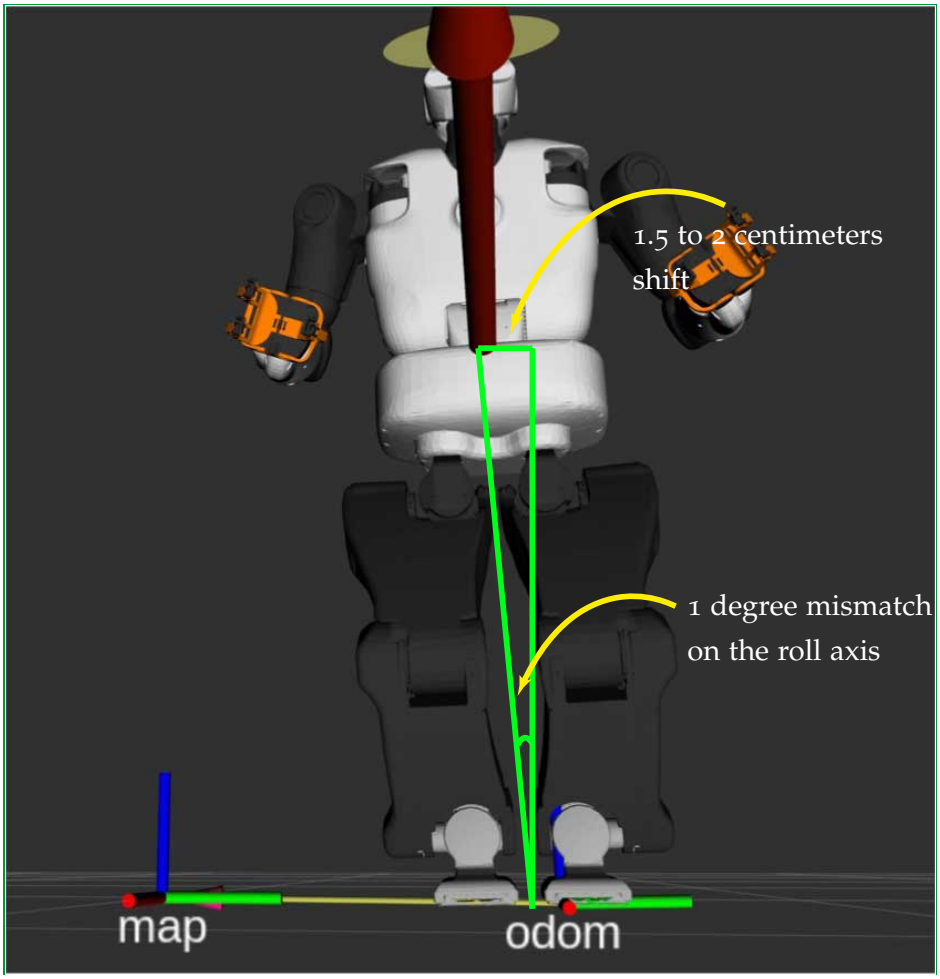


Figure 36: The mis-alignment between the planning frame and the odometry frame can tilt the motion plan in the odometry frame. This image shows an example when the planning frame and the odometry frame have about 1-degree orientation mismatch around the positive roll axis. This orientation mismatch causes the CoM to have about 1.5 to 2 centimeter shift towards the right foot. When the robot starts walking with this titled posture, the robot can fall when the right foot breaks the contact (as the tilt brings the CoM away from the support region of the left foot).

Although relying on VICON to correct the state estimation drift sounds promising, the usage of VICON also constrains the robot to work inside a lab environment that has to be fully covered by the VICON infrared cameras. Nevertheless, when

considering real-world applications (for example, exploring unknown environments), relying on external sensing to correct the state feedback is un-realistic. In order to handle real-world scenarios, the robot necessarily requires the capability to achieve accurate state feedback through the fusion of on-board sensors. Additionally, in the real world, the state estimator also needs to capture the abrupt state deviations caused by foot slippage and external force disturbance, which can be a challenging task. To enable real-world operation, we believe further research on the state estimation of the legged robot is necessary.

8.3.2 *Footstep Tracking Error*

Apart from the state estimation issue, we also find that the whole-body inverse dynamics controller can have errors in the footstep tracking. For example, when the robot creates a new contact, the foot yaw can have around 1.5 degrees divergence from the reference (0 degrees). Moreover, compared to the planned footstep location, the actual footstep location of the robot can also have a 1-2 centimeters deviation along the y-axis. These deviations on the footsteps can cause the robot to lose its balance during walking. For instance, in the presence of these footstep deviations, the desired CoM trajectories may pass the border of the stance foot during the single support phase. In that case, the robot may not have sufficient force to support its weight, and the whole-body controller may fail to update a feasible solution. Then, the robot will fail to maintain its balance and fall.

To mitigate this issue, we choose to assume a smaller foot size during our planning stage. With this smaller foot size, the planner can generate more conservative motion plans, as the robot will tend to support the CoM with the interior part of the foot during the single support phase. This can make the motion plan more resilient to the footstep divergence issue, i.e., during the motion execution, even if the footstep diverges from the planned locations, the desired CoM trajectory can still be supported by the interior part of the foot.

To achieve more robust operation in the real world, we believe it is necessary to improve the footstep tracking performance of the whole-body controller. To approach this aim, we suggest to consider the following two aspects. First, we recall that the swing foot trajectory is interpolated after we compute the motion plan of the robot, and we assume that the swing foot trajectory will generate zero momentum. However, given the large inertia and mass of the robot legs, it is unlikely that the swing foot trajectory can create zero momentum. As a result, the CoM tracking task and the swing foot tracking task may have conflicts with each other, and the whole-body controller has to find a compromise between the tracking of these two tasks. To address this

issue, we suggest to switch the whole-body controller to a MPC controller based on DDP [62]. This MPC controller can online adapt the swing foot trajectory in every control loop while considering the whole-body dynamics of the robot. We believe this MPC controller can improve the tracking performance of the footsteps. Furthermore, it is also important to note that the control techniques we have are all model-based. As a consequence, the tracking performance of these controllers is dependent on the accuracy of the robot model. To improve the footstep tracking performance, it may be beneficial to further refine the model of the robot, i.e., provide a more accurate measurement of the mass and inertia of each link. This can be achieved by a more precise system identification process [19].

8.4 CONCLUSION

In this chapter, we have presented our real-world experiments on the humanoid robot Talos to demonstrate the effectiveness of our locally-guided RHP approach in achieving online multi-contact receding horizon planning on uneven terrain and environments with dynamic changes. We show that the fast computation of locally-guided RHP can enable our robot to online adapt its motions according to the latest terrain condition perceived by the robot. This can allow the robot to safely traverse the terrains that change on-the-fly. Furthermore, we also discussed the challenges we met in real-world experiments, i.e., state estimation drift and footstep tracking errors. We described these issues and discussed the future directions to address them. We believe further research in state estimation and control of legged robots is also important for further improving the performance of the legged robots.

9

AUTOMATIC GAIT PATTERN SELECTION FOR LEGGED ROBOTS

In this chapter, we present our exploration of addressing the gait pattern selection problem of legged robots.

When synthesizing legged locomotion plans, an important issue is the combinatorial complexity that arises from gait pattern selection. Though it can be defined manually, the gait pattern plays an important role in the feasibility and optimality of a motion with respect to a task. Replacing human intuition with an automatic and efficient approach for gait pattern selection would allow more autonomous robots, responsive to task and environment changes. To this end, we propose the idea of building a map from task to gait pattern selection for given environment models and performance objectives. We carry out case studies for a 2D half-cheetah model and a quadruped robot. We show that we can establish direct mappings between a given task and an optimal gait pattern for these legged models. Furthermore, to validate the trajectories planned by our method, we perform real-world robot experiments on the ANYmal robot.

We organize this chapter as follows. In Section 9.1, we provide an overview of the gait pattern selection problem as well as our method and findings. Next, in Section 9.2, we present a formal description of the gait pattern selection problem. Then, in Section 9.3, we describe our technical approach for building the gait pattern selection map. Afterwards, we present our experiment result on a 2D half-cheetah model and the 3D quadruped robot ANYmal. Lastly, in Section 9.5, we conclude the chapter.

9.1 INTRODUCTION

Motion and contact planning for legged locomotion in arbitrary environments is an open problem. In particular, optimizing the gait pattern adds combinatorial complexity to this high-dimensional problem. This complexity can be interpreted as a wide range of possible motions for each gait pattern. However, an appropriate gait pattern selection might be crucial to find a feasible motion [2, 80]. To avoid this challenge, most of the traditional approaches require to pre-specify the gait pattern, e.g. [6, 16, 46, 62, 63, 77, 94, 110]. However, biological and computational studies have identified that the choice of gait pattern has a large impact on the performance of legged locomotion [83,

88, 111, 113]. These studies show that gait pattern selection highly depends on the locomotion speed and task requirement. Furthermore, some of the gait patterns might not be suitable choices in a given terrain condition, e.g., climbing [71], and manually defining appropriate contact sequences is tedious and difficult.

To enhance autonomous operation of legged robots while allowing fast computation of legged locomotion plans with varying gait patterns, in this chapter, we aim to build a map from task conditions to optimal gait patterns. The main idea is to encode this map such that we can quickly retrieve the optimal gait pattern during online computation of legged locomotion plans, i.e., the online multi-contact RHP approaches presented in Chapter 4 to Chapter 8, or other multi-contact planners. In this exploratory work, we represent robots using single rigid body models. The scalability of the approach towards more complex robot models is discussed in the last section.

Findings and method overview

We establish optimal gait pattern maps for a 2D half-cheetah model and the 3D quadruped robot ANYmal [43] in a defined task descriptor space with given environment models and performance objectives. Across this chapter, we use the term *optimal* to refer to the locally optimal solutions we found by our nonlinear optimization-based approach enhanced with multiple random restarts. Through our investigation, we find that:

- The maps typically form several contiguous regions for which a particular gait pattern is optimal. This property allows us to use supervised learning to capture the structure of the map efficiently.

- The optimal trajectories within each gait region are qualitatively similar, which implies that they are in the same basin of attraction. We empirically find that the mean of these trajectories can be used to initialize an optimization-based planning algorithm. By utilizing the optimal gait pattern maps along with the obtained initial seeds, we achieve significant gains in computation time for our trajectory optimization method, e.g., improving the computation time from hours or minutes to 0.1-0.3 seconds.

To generate the map, we propose a nonlinear optimal control formulation based on Mixed-Integer Non-Linear Programming (MINLP). With our formulation, we offline compute a dataset of optimal gait patterns and state trajectories for a predefined set of tasks with given environment models and performance objectives. We describe the

task by the locomotion speed and cycle duration, and the environment by its slope angle. The performance objective is defined as the cost function of the state and control trajectories. After computing the date set, we employ a neural network to learn and encode the map.

To validate the dynamic feasibility of the trajectories planned by our method, we also perform real-world robot experiments on the ANYmal quadruped.

9.2 PROBLEM DESCRIPTION

In this section, we formulate the problem of finding the optimal gait pattern selection map.

To give more detail, we model the robot as a single rigid body along with N_L mass-less legs (See Fig. 37). While this assumption may sound limiting for hardware execution, this is less likely to affect the decisions on the gait pattern selection. Note that we could still rely on instantaneous or predictive controllers [65, 73, 114] to locally adapt the motions while considering the leg inertia.

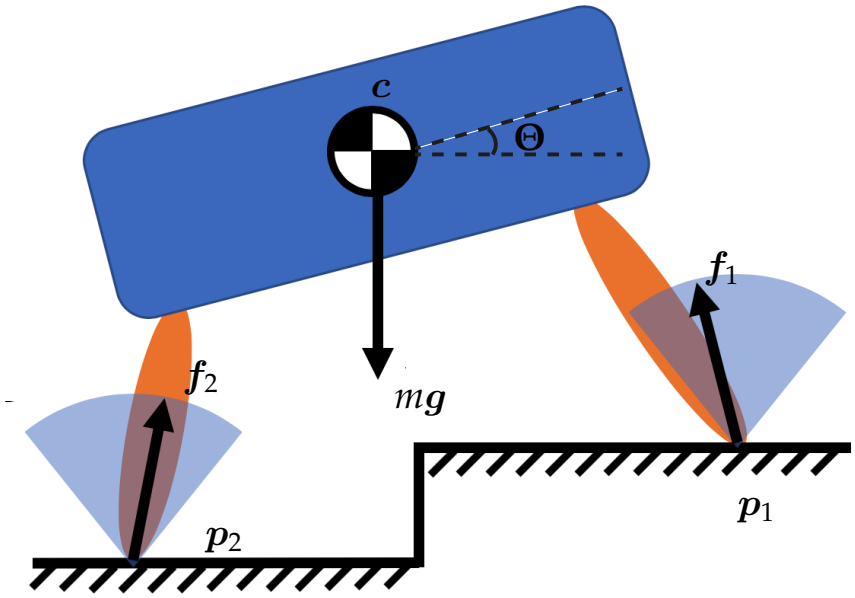


Figure 37: We model the robot as a single rigid body with mass-less limbs. We use $\boldsymbol{c} \in \mathbb{R}^3$ to denote the CoM position of the base, and $\boldsymbol{\Theta} \in \mathbb{R}^3$ to represent the base orientation modeled by Euler angles. We denote by m as the mass of the robot and $\boldsymbol{g}$ as the gravitational acceleration. Moreover, we use $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ to represent the locations of the feet, and define $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ as the contact force vectors subject to friction cone (light blue regions) constraint.

To model the robot, we define $\boldsymbol{c} \in \mathbb{R}^3$ as the CoM position of the base, and $\boldsymbol{\Theta} \in \mathbb{R}^3$ as the base orientation represented by Euler angles (roll, pitch, and yaw angles). Furthermore, we denote by $\boldsymbol{p}_l \in \mathbb{R}^3, l \in \{1, ..., N_L\}$ as the location of the l-th foot, and we use $\boldsymbol{f}_l \in \mathbb{R}^3$ to denote the contact force vector of the foot $l \in \{1, ..., N_L\}$ (assume each foot as a point-contact).

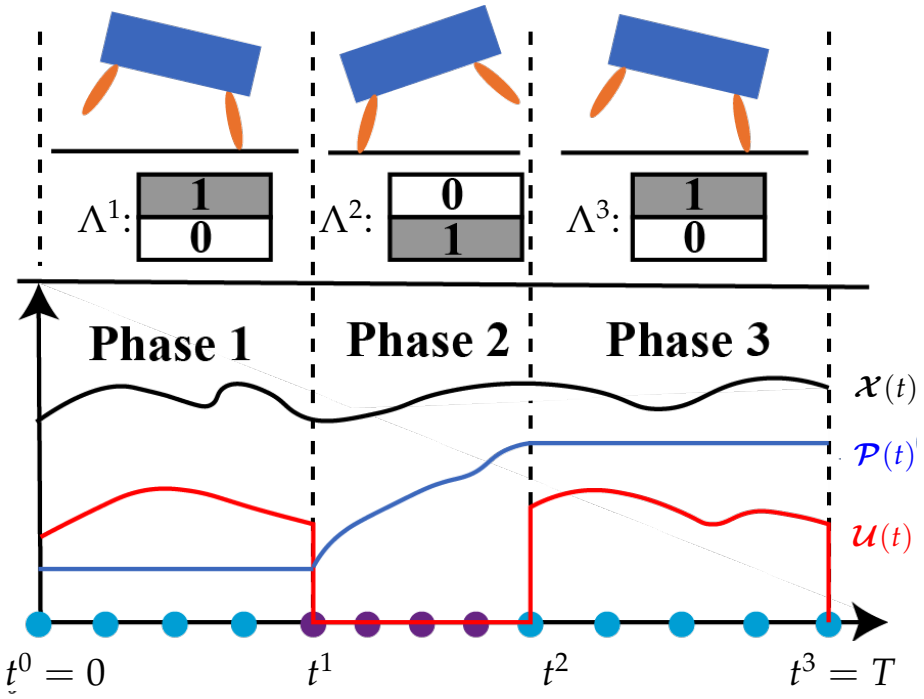


Figure 38: Example of the gait pattern and the motion plan. $\mathcal{X}(t)$, $\mathcal{P}(t)$ and $\mathcal{U}(t)$ refer to the base trajectory, feet trajectories, and control trajectories (contact force profiles), respectively. For illustration purposes, we use scalars to represent these vectors. The motion is described using N_{ph} phases. Each phase has an associated contact configuration Λ^q and switching time t^q. Here we show an example gait pattern where 0 and 1 represent active and inactive contact, respectively. Different gait patterns can be represented by toggling the binary values of Λ^q. The time t is discretized using a variable time-step parameterization, where each phase has the same number of integration nodes (blue and purple dots) that scale proportionally with respect to the switching time t^q.

We denote by a scenario as $\boldsymbol{\xi} = [\boldsymbol{\kappa}_{task}, J, \boldsymbol{\Omega}]^T$, where $\boldsymbol{\kappa}_{task}$ is the task descriptors, J is the performance objective—a cost function on the state and control, and Ω is the environment model. We consider periodic locomotion task where the task descriptor $\boldsymbol{\kappa}_{task} = [\boldsymbol{v}, T]$ defines the forward locomotion speed v of the CoM moving along the tangential direction of the environment and the cycle duration T. The performance objective J is defined as a user-specified cost function on the robot state and control

vectors. Furthermore, we assume the environment $\boldsymbol{\Omega}$ is a slope with the inclination defined by γ.

Given a scenario specification $\boldsymbol{\xi}$, our aim is to compute a motion plan $\boldsymbol{\pi}(t)$ for a legged robot, which contains the state trajectory $\boldsymbol{\mathcal{X}}(t)$, the control trajectory $\boldsymbol{\mathcal{U}}(t)$ and the foot trajectory $\boldsymbol{\mathcal{P}}(t)$:

$$\boldsymbol{\pi}(t) = [\boldsymbol{\mathcal{X}}(t), \boldsymbol{\mathcal{U}}(t), \boldsymbol{\mathcal{P}}(t)], \tag{27}$$

for a finite time horizon $t \in [0, T]$. We define that the state trajectory $\boldsymbol{\mathcal{X}}(t) = [\boldsymbol{c}(t), \boldsymbol{\Theta}(t)]$, which is composed of the base CoM trajectory $\boldsymbol{c}(t)$ and the base orientation trajectory $\boldsymbol{\Theta}(t)$. Furthermore, we also define that the control trajectory $\boldsymbol{\mathcal{U}}(t) = [\boldsymbol{f}_1(t), ..., \boldsymbol{f}_{N_L}(t)]$, which collects the contact force profiles of all feet $l \in \{1, ..., N_L\}$. Similarly, we denote the feet trajectory as $\boldsymbol{\mathcal{P}}(t) = [\boldsymbol{p}_1(t), ..., \boldsymbol{p}_{N_L}(t)]$ which includes the trajectories of all feet $l \in \{1, ..., N_L\}$.

We recall that, as mentioned in Chapter 3, the feet trajectories $\boldsymbol{\mathcal{P}}(t)$ and control trajectories (contact force profiles) $\boldsymbol{\mathcal{U}}(t)$ of a legged robot are discontinuous at the instants of contact transitions, i.e., a foot breaks or makes contacts (see the example in Fig. 38). Such discrete events divide the motion plan into N_{ph} phases, where in each phase, the contact configuration is a constant. We model the contact configuration for each phase $q \in \{1, ..., N_{ph}\}$ with binary variables $\Lambda^q \in \{0, 1\}^{N_L}$, which indicates the contact states of each leg in phase q. For example, 1 means a foot is in contact with the ground, while 0 means a foot is detached from the ground. As we can observe from Fig. 38, these contact configurations capture the underlying structure of the motion plan. In locomotion, we call such a structure as gait pattern, defined as:

$$\boldsymbol{\Lambda} = [\Lambda^1, ..., \Lambda^{N_{ph}}]. \tag{28}$$

When planning motions for legged robots, reasoning the gait pattern selection introduces combinatorial complexity, which can result in expensive computation. As a result, existing approaches often rely on human intuition to supervise the gait pattern selection. However, this can limit the autonomy of the robot, and selecting a gait pattern for challenging environments can be tedious and difficult. To enhance the autonomy of legged robots, while allowing efficient computation of legged locomotion plan, we propose to leverage offline computation to build a direct mapping from task specifications $\boldsymbol{\kappa}_{task}$ to the optimal gait pattern for a given environment Ω and performance objective J:

$$\mathbf{\Lambda}^*_{\Omega,J} = \mathbf{g}_{\Omega,J}(\boldsymbol{\kappa}_{task}), \tag{29}$$

To build the map, we use a supervised learning approach to learn the mapping from an offline computed data-set. This data-set demonstrates gait pattern selection for various of tasks considering different environment models and performance objectives. To obtain the data-set, we use MINLP to discover the gait pattern for different task samples. Next, we describe our technical approach in detail.

9.3 TECHNICAL APPROACH

9.3.1 *Overview*

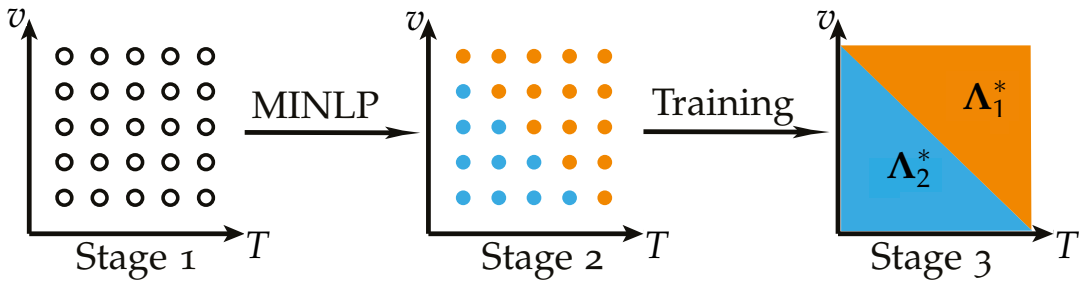


Figure 39: Procedure for building the optimal gait pattern selection map. Given an environment model and performance objective, we firstly uniformly sample the task specification space (Stage 1). We then use our MINLP approach to discover optimal gait patterns for these task samples (Stage 2). Eventually, we use the pairs of task samples and discovered optimal gait patterns to construct a training set and train a neural network to learn the gait pattern selection policy (Stage 3).

As Fig. 39 indicates, we take a supervised learning approach to build the optimal gait pattern selection map for given environment models and performance objectives. The key component of our approach is a novel MINLP formulation that can compute optimal gait patterns for a chosen task specification. We use this MINLP approach to compute optimal gait patterns for a couple of task samples. These task samples and associated optimal gait patterns are then used as a training data set to train a neural network classifier to capture the optimal gait pattern selection policy. We introduce the

details of our MINLP approach in Section 9.3.2 and the supervised learning approach in Section 9.3.3.

9.3.2 *Discovering Optimal Gait Pattern with Mixed-Integer Non-Linear Programming*

In this section, we introduce our Mixed-Integer Non-Linear Programming (MINLP) formulation for discovering optimal gait patterns. Our approach computes the optimal gait pattern $\mathbf{\Lambda}^*$ by solving a mixed-integer optimal control problem with N_{ph} phases.

Decision Variables

The decision variables of our optimal control problem include:

- The motion plan $\boldsymbol{\pi}(t)$ consists of the base trajectory $\boldsymbol{\mathcal{X}}(t)$, control trajectory (contact force profiles) $\boldsymbol{\mathcal{U}}(t)$, as well as the feet trajectories $\boldsymbol{P}(t)$.
- The gait pattern matrix $\mathbf{\Lambda} \in \{0,1\}^{N_L \times N_{ph}}$.
- The phase switching timings $\boldsymbol{\mathcal{T}} = [t^1, ..., t^{N_{ph}}]$, where $t^q \in \mathbb{R}$ denotes the terminal time of q-th phase (Fig. 38).

Mixed-integer Optimal Control Formulation

We formulate the mixed-integer optimal control problem as follows:

$$\min_{\boldsymbol{\pi}, \mathbf{\Lambda}, \boldsymbol{\mathcal{T}}} \quad \sum_{q=1}^{N_{ph}} \int_{t^{q-1}}^{t^q} J(\boldsymbol{\pi})\, dt \tag{30a}$$

$$\begin{aligned}
\text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{\pi}_0, \boldsymbol{\pi}_T, \boldsymbol{\kappa}_{\text{task}}) \leq 0, && \text{(task)} && \text{(30b)}\\
& 0 = t^0 \leq t^1 \leq \cdots \leq t^{N_{ph}} = T, && \text{(timings)} && \text{(30c)}\\
& \boldsymbol{\mathcal{F}}_{dyn}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{P}}) = 0, && \text{(dynamics)} && \text{(30d)}\\
& \boldsymbol{\mathcal{P}} \in \boldsymbol{B}(\boldsymbol{\mathcal{X}}), && \text{(kinematics)} && \text{(30e)}\\
& \text{For } t \in [t^{q-1}, t^q]: && && \\
& \quad \boldsymbol{h}_{\Lambda^q}(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{P}}, \boldsymbol{\Omega}) \leq 0, && \text{(contacts)} && \text{(30f)}
\end{aligned}$$

where we use $J(\cdot)$ to denote the cost function, and $\boldsymbol{h}$ to represent a generic inequality constraint. For simplicity, we do not explicitly write the time dependency for the decision variables.

Constraints

To ensure the motion plan is physically consistent, we define different constraints in (30b)-(30f). Next, we elaborate their formulations in detail:

1) Task:

In (30b), we impose the periodic task constraints based on the task specification $\boldsymbol{\kappa}_{task}$:

$$\boldsymbol{c}_T = \boldsymbol{c}_0 + \boldsymbol{v}T, \tag{31a}$$

$$\boldsymbol{\Theta}_T = \boldsymbol{\Theta}_0, \tag{31b}$$

$$\boldsymbol{\mathcal{P}}_T = \boldsymbol{\mathcal{P}}_0, \tag{31c}$$

where the terminal base CoM state $\boldsymbol{c}_T$ is enforced to travel a distance of $\boldsymbol{v}T$ with respect to the initial base CoM state $\boldsymbol{c}_0$. Furthermore, the periodic task constraints also enforce the terminal base orientation $\boldsymbol{\Theta}_T$ and the terminal foot locations $\boldsymbol{\mathcal{P}}_T$ to return to their initial values $\boldsymbol{\Theta}_0$ and $\boldsymbol{\mathcal{P}}_0$.

2) Timings:

We introduce in-equality constraints (30c) to ensure the phase switching times t^q are increasing monotonically. Additionally, we constrain the total motion duration (defined by $t^{N_{ph}}$) to be the same as the cycle duration given by T.

3) Dynamics:

In (30d), we impose the system dynamics constraint using the Newton-Euler equations for a single rigid body:

$$m\ddot{\boldsymbol{c}} = m\boldsymbol{g} + \sum_{l=1}^{N_L} \boldsymbol{f}_l \tag{32a}$$

$$\mathcal{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathcal{I}\boldsymbol{\omega}) = \sum_{l=1}^{N_L} \boldsymbol{f}_l \times (\boldsymbol{c} - \boldsymbol{p}_l), \tag{32b}$$

where m is the total mass, $\mathcal{I}$ is the inertia tensor of the base of the robot, $\boldsymbol{\omega}$ is the angular velocity transformed from Euler angles and their rates, $\boldsymbol{g}$ is the gravitational acceleration.

Moreover, contact forces $\boldsymbol{f}_l$ is further constrained with the friction cone constraint:

$$-\mu \boldsymbol{f}_l^{\hat{n}} \leq \boldsymbol{f}_l^{\hat{t}_1,\hat{t}_2} \leq \mu \boldsymbol{f}_l^{\hat{n}} \tag{33}$$

where μ is the friction coefficient, $\boldsymbol{f}_l^{\hat{n}}$ is the normal component of the contact force, and $\boldsymbol{f}_l^{\hat{t}_1}$, $\boldsymbol{f}_l^{\hat{t}_2}$ are the tangential components of the contact force.

4) Kinematics:

In (30e), we approximate the reachable space of each leg with a box placed at the default foot position $\tilde{\boldsymbol{p}}_l$:

$$|\boldsymbol{R}\,(\boldsymbol{p}_l - \boldsymbol{c}) - \tilde{\boldsymbol{p}}_l| \leq \boldsymbol{b}, \tag{34}$$

where the $\boldsymbol{b}$ is the half length of the box, and $\boldsymbol{R}$ is the transformation matrix from the world frame to the robot base frame based on the current state of the base.

5) Contacts:

The contact dynamics constraint (30f) is defined as follows:

$$\begin{aligned}
& \boldsymbol{f}_l^{\hat{n}} \geq 0, && \text{(unilaterality)} && \text{(35a)}\\
& \boldsymbol{\sigma}(\boldsymbol{p}^l, \boldsymbol{\Omega}) \geq 0, && \text{(non-penetration)} && \text{(35b)}\\
\Lambda_l^q = 0 \Rightarrow\ & \boldsymbol{f}_l = 0, && \text{(inactive contact)} && \text{(35c)}\\
\Lambda_l^q = 1 \Rightarrow\ & \boldsymbol{\sigma}(\boldsymbol{p}_l, \boldsymbol{\Omega}) = 0,\ \dot{\boldsymbol{p}}_l = 0, && \text{(active contact)} && \text{(35d)}
\end{aligned}$$

where $\boldsymbol{\sigma}(\boldsymbol{p}_l, \boldsymbol{\Omega})$ measures the clearance between the foot and the environment, and Λ_l^q denotes the contact configuration of a leg l at phase q. This set of constraints enforces:

- Unilateral contact forces since the robot can only push against the environment (35a).
- The foot cannot penetrate the ground (35b).

- If the contact state is 0 (inactive contact), the contact force should be zero (35c).

- If the contact state is 1 (active contact), the distance between the foot and the environment should be zero (the foot should stay in contact with the ground), along with zero foot velocity (35d).

Note, the constraints (35c) and (35d) describe the complementary condition of contacts [81] through an integer representation, and we use the standard big-M formulation to model them [112].

Transcription to a Mixed-Integer Nonlinear Programming Problem

To solve the presented optimal control problem (30a) numerically, we transcribe it into a MINLP problem by using a variable time-step parameterization. As Fig. 38 illustrates, each phase has the same number of uniformly distributed knots whose time interval scales proportionally to the phase switching timings t^q, and we impose the constraints (30b)-(30f) on each knot. Furthermore, we approximate the integral such as the system dynamics constraint and the cost function with a forward Euler integration scheme.

9.3.3 *Learning the Gait Pattern Selection*

To obtain the gait pattern selection map for a chosen environment model $\mathbf{\Omega}$ and a performance objective J, we firstly pick D tasks $\{\boldsymbol{\kappa}_{task,d} = [v_d, T_d]^T\}$ that are uniformly sampled in the task descriptor space as illustrated in Fig. 39. Then, we compute the optimal gait patterns $\mathbf{\Lambda}_d^*$ of these selected task samples $\boldsymbol{\kappa}_{task,d}$ using our MINLP approach. Since our approach can only guarantee local optimality due to the nonlinear nature of the MINLP problem, we run the optimization 50 times for each sampled task $\boldsymbol{\kappa}_{task,d}$ with randomized initial seeds to increase the chances of finding the global minimum. We use the pairs of task samples and optimal gaits $\{\boldsymbol{\kappa}_{task,d}, \Lambda_d^*\}$ as the training data set, and we train a neural network classifier to learn the optimal gait pattern selection policy from the generated data set. Our neural network has 3 hidden layers with a total number of 230 neurons. The implementation and training of the neural network are achieved by scikit-learn [78].

In this section, we present our result for establishing the optimal gait pattern maps for different terrains and performance objectives. Meanwhile, we also highlight their properties and explain how they can be integrated into existing TO frameworks to gain performance improvements.

9.4.1 *Implementation Details*

In all cases, we model the optimization problems using MATLAB while using the branch-and-bound algorithm provided by KNITRO 10.10 [12] to solve the MINLP problems. Furthermore, we provide the gradient and Hessian using the automatic differentiation framework CasADi [3]. The dataset for training the neural network is computed by the cluster Eddie provided by the Edinburgh Compute and Data Facility (ECDF). For online evaluations, we use a computer with an Intel Xeon E3-1535M v6 (Maximum 4.20 GHz) and 32 GB memory.

9.4.2 *Gait Pattern Discovery for a 2D Half-Cheetah Model*

First, we consider a 2D half-cheetah model. We compare the results for different terrain inclinations γ and performance objectives J. For all cases, we set the number of phases to 4, with 10 knots per phase. Even though we use a simplified model and a small number of phases, there are a total of $4^4 = 256$ integer combinations for the optimizer to consider. To avoid undesirably fast phases, we constrain each phase duration to at least 10% of the cycle duration T.

9.4.2.1 *Contact forces minimization*

We start by considering a flat terrain ($\gamma = 0$) and a performance objective of minimizing contact forces:

$$J_1 = \int_0^T \sum_{l=1}^{N_l} \boldsymbol{f}_l^2 \, dt. \tag{36}$$

This cost can be interpreted as the minimum effort to achieve the desired task.

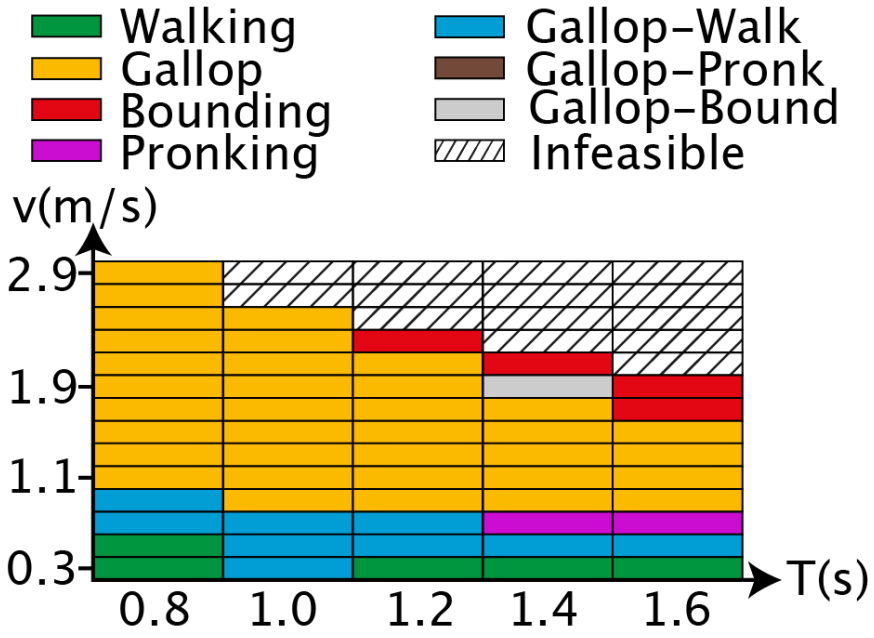


Figure 40: Optimal gait patterns for a 2D half-cheetah model on flat terrain with performance objective J_1.

The optimized gait patterns are summarized in Fig. 40, while all discovered gait patterns are described in Fig. 41. At very low speeds (up until $0.3m/s$ approximately), the model favors a walking gait. If we slightly increase the speed (0.5 to $0.7m/s$), the model switches to a gallop-walk gait. If we further increase the speed ($0.9m/s$ and onward), galloping is preferred. For large cycle durations (1.2 to $1.6s$) and speeds near the maximum (1.7 to $2.3m/s$), bounding is preferred over galloping. Finally, pronking and gallop-bounding gaits cover two small regions only, the former during the transition from gallop-walk to gallop, while the latter during the transition from galloping to bounding.

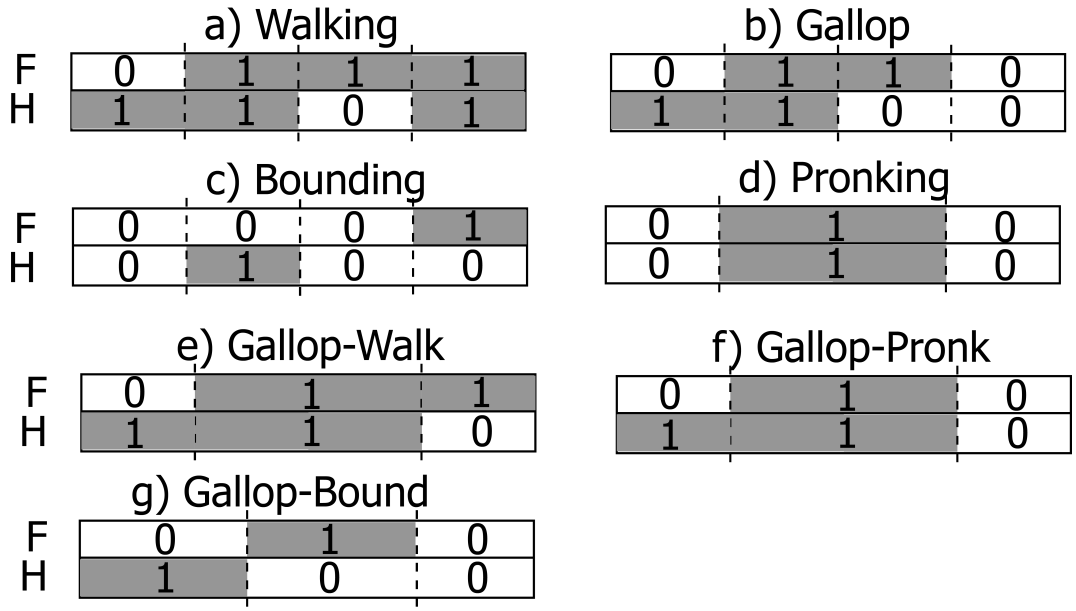


Figure 41: Discovered gait patterns for the 2D half-cheetah model. F refers to front and H to hind leg, 0 to inactive and 1 to active contacts. We categorize the gait patterns (e), (f), and (g) into the gallop family, as they are asymmetric gait patterns that can be derived from the gallop gait (b). For example, by removing the flying phase of the gallop gait, we can obtain the gallop-walk gait (e). Furthermore, by vanishing the front-support phase of the gallop gait, we can achieve the gallop-pronk gait (f). Lastly, if we remove the double support phase of the gallop gait, we can obtain the gallop-bound gait (g). The motions for some of these gait patterns can be found https://youtu.be/ylfYnl7sZDg.

9.4.2.2 *Trunk vibration minimization*

We switch the performance objective to:

$$J_2 = \int_0^T ((v_t - v)^2 + v_n^2 + (\theta - \gamma)^2 + \dot{\theta}^2)\, dt, \tag{37}$$

where v_t and v_n are the tangential and normal CoM velocities with respect to the terrain, and θ is the pitch angle. This cost encourages to minimize robot base vibration. This is a common requirement during inspection missions, i.e., the camera on the robot needs to remain fixed on a point of interest. Since the terms in (37) can be infinitesimally small (approach 0), we scale all of them by 10^6 times to increase the numerical stability.

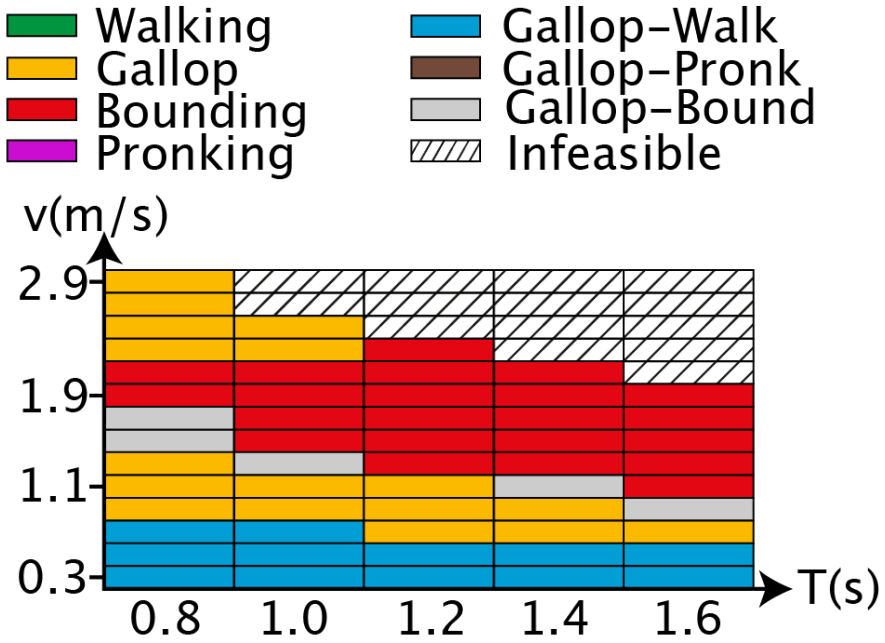


Figure 42: Optimal gait patterns for a 2D half-cheetah model on flat terrain with performance objective J_2.

The optimized gait patterns are shown in Fig. 42. We observe that some regions of optimal gaits remain the same. However, there are some salient changes worth discussing. For medium velocities and across the entire cycle duration range, the bounding gait is preferred over the galloping one. This also leads to an increase in the gallop-bound transient gait at the borders of the two regions. Additionally, the gallop-walk gait becomes prevalent at the low-speed regime.

9.4.2.3 *Trunk vibration minimization on a 20° slope*

We set the terrain slope to 20° and use the performance objective J_2. Fig. 43 summarizes the results for this case. We observe that bounding gaits are replaced by galloping gaits for low cycle duration in the medium velocity regime. Furthermore, pronking gaits become more prominent on the boundary between these two gaits.

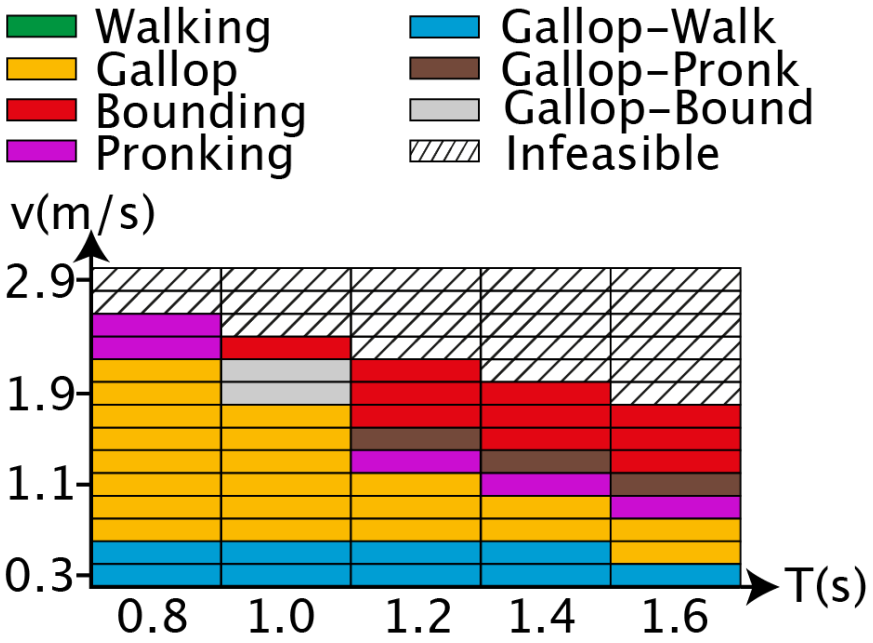


Figure 43: Optimal gait patterns for a 2D half-cheetah model on 20° sloped terrain with performance objective J_2.

9.4.2.4 *Discussion*

Depending on the optimized performance objective and environment characteristics, optimal gait pattern selection can have sufficient differences. As a result, it is unreasonable to expect a human operator to have the intuition to select proper gait patterns online during motion planning. Our approach provides a precise and systematic way to capture these differences, making automatic gait selection possible without needing to compute gaits every time from scratch—albeit for a particular robot dynamics.

Another important observation is that *optimal gait patterns exhibit structural properties*. For example, distinctive gait patterns tend to occupy similar regions, as shown on the discovered maps (Figs. 40, 42 and 43). This observation motivates the use of machine learning techniques to capture the relationships among similar task descriptors, without resorting to dense indexing, i.e., the simple strategy described in Section 9.3.3, which clusters the optimal gait patterns using a neural network.

9.4.3 *Fast Computation of Locomotion Plans*

In this section, we explore the following question: are there substantial differences between the state trajectories within similarly clustered task descriptor samples?

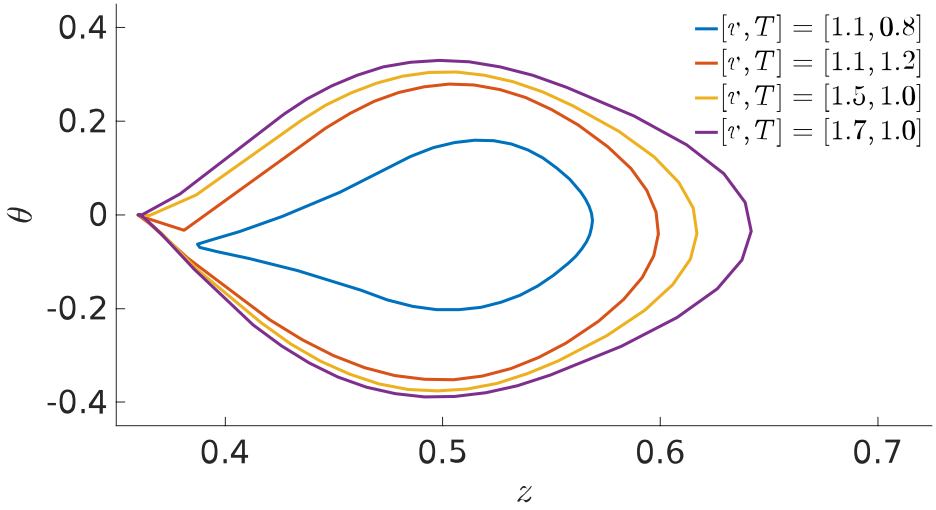


Figure 44: Example state-space trajectories for the gallop gait region. θ is the pitch angle of the base in radius, and z is the CoM height of the base in meters.

And if not, how can we exploit this property? In our findings, we encounter that the trajectories do not exhibit large differences. In Fig. 44 and Fig. 45, we plot the state-space trajectories for the CoM height z and body orientation θ for two different gait patterns. The resulting trajectories look qualitatively similar. It seems that each gait region has its own basin of attraction. Using this property, we can build simple strategies to warm-start the TO problems that compute state trajectories when the gait pattern is fixed by our gait pattern selection map. Concretely, for each optimal gait pattern region classified by the neural network, we estimate a nominal initial trajectory, e.g., the mean of the sampled trajectories.

Using this strategy, we perform a comparison between two TO frameworks with pre-specified gait patterns. For both methods, we use the neural network to select the appropriate gait pattern given the task descriptor. For the first method, we randomly initialize the state variables. For the second method, we identify the region where the gait pattern belongs to, and use the nominal initial trajectory as the initial seed. We solve the resulting TO problems for 10 random task descriptors for all 3 case studies described in Section 9.4.2.1, Section 9.4.2.2 and Section 9.4.2.3. Thus, we run each method for 30 times. For both MINLP and TO we set 40 knots, which results in 1779 constraints for both methods. Furthermore, the MINLP has 742 variables, while the TO has 734.

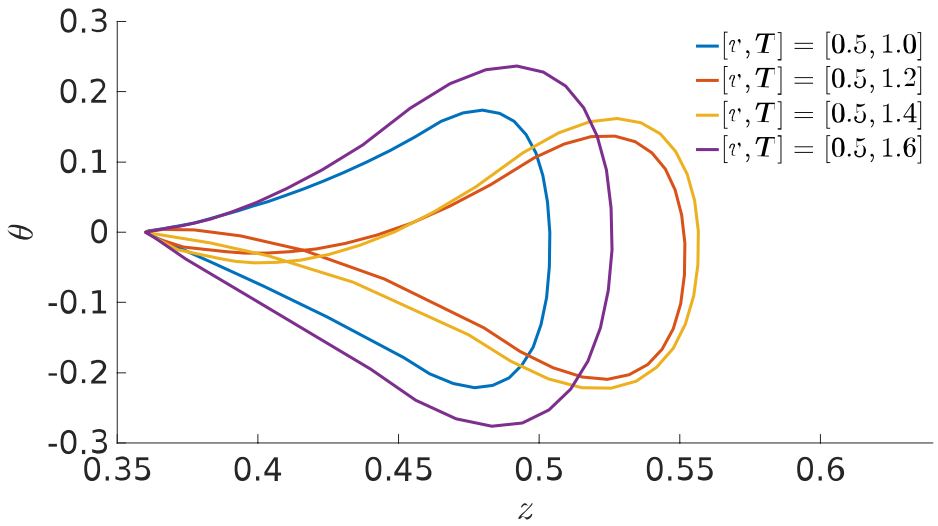


Figure 45: Example state-space trajectories for the gallop-walk gait region. θ is the pitch angle of the base in radius, and z is the CoM height of the base in meters. From the figure, we can observe that the trajectories can have self-intersections. We speculate that these self-intersections can result from the cost function we have chosen, which aims to minimize contact forces. When possible, this cost function encourages the optimizer to generate motions with smooth momentum changes (accelerations). Consequently, when the motion involves reversing the momentum along a specific axis, such as changing the base from a titling-down posture (negative pitch) to a tilting-up posture (positive pitch), the robot tends to favor a gradual turning in the state trajectory (resulting in self-intersection) rather than an abrupt change.

We summarize in Table 7 the results of this comparison. First, we report the results for the MINLP approach. We use them as a baseline to compute the consistency of the compared methods. Consistency is defined as the number of trials that the optimizer converged to the same solution as in the MINLP case. Additionally, we report the successful runs, which correspond to the number of trials that the optimizer was able to output a valid locally optimal solution. There are two reasons that the optimizer might fail: either by exceeding the maximum number of iterations (10000) or by failing to output a feasible solution.

As we can see from Table 7, the MINLP approach takes a considerable amount of time to compute (it needs averagely 93s to converge). This is expected as the MINLP has to address the combinatorial aspect of the planning problem. When we fix the gait pattern with the optimal gait pattern selection maps, we observe significant computation time gains, where the average computation time drops to 3s. By using the nominal initial trajectory to warm-start the optimizer, we further decrease the

average computation time to 115ms. Furthermore, using the nominal initial trajectory provides additional improvements. For instance, we increase the chances that the TO converges to a feasible local minimum, and the TO can also discover more consistent local minima with respect to MINLP results.

Table 7: Computation performance between the MINLP and the two TO methods for computing the motion plan for the 2D half-cheetah model with pre-specified optimal gait pattern, with and without nominal state initialization. We test 10 randomly sampled task specifications for the each of the 3 case studies described in Section 9.4.2.1, Section 9.4.2.2 and Section 9.4.2.3.

Method	Computation time (s)	Successful Convergence	Consistency
MINLP	93.66±53.7	30/30	N/A
TO (Gait pattern only)	3.05±3.26	25/30	13/30
TO (Gait pattern & warm-start)	**0.115±0.02**	30/30	30/30

9.4.4 *Optimal Gait Pattern Discovery for the 3D Quadrupedal Robot ANYmal*

We repeat the previous analysis for the 3D quadruped robot ANYmal. Our aim here is to demonstrate scalability to more complex models, and investigate empirically whether the previous conclusions can be extended to a 3D case. To demonstrate a wide range of motions, we allow large contact forces (max. $500N$) and foot velocity (max. $2.5m/s$).

We select flat terrain and a performance objective that penalizes simultaneously contact forces, trunk vibration, and lateral feet displacements with respect to the nominal position:

$$J_3 = \int_0^T \left(w_1 \boldsymbol{F}^2 + w_2 (v_t - v)^2 + w_2 v_n^2 \right. \\ \left. + w_2 \boldsymbol{\Theta}^2 + w_2 \dot{\boldsymbol{\Theta}}^2 + w_2 (\boldsymbol{P}_y - \bar{\boldsymbol{P}}_y)^2 \right) dt, \tag{38}$$

where $\boldsymbol{P}_y$ is the lateral feet displacements, $\bar{\boldsymbol{P}}_y$ is the nominal feet displacements, $w_1 = 10^{-2}$ and $w_2 = 10^6$. Additionally, we define the lower bound on each phase duration as 10% of the cycle duration T.

To model all possible gait pattern combinations of the quadruped robot, we would need 8 phases. This makes the problem challenging for the MINLP in the current form, since there are $16^8 = 4\,294\,967\,296$ possible combinations. One way to handle this is by

introducing convex relaxations as in [2, 80], which would allow us to use mixed-integer convex optimization. This would remove the need to solve MINLP with multiple randomized initial seeds; albeit increasing the problem size due to the introduction of auxiliary integer variables [2, 80] needed to select convex pieces of nonlinear functions, but could saving up time for large dimensional problems. Alternatively, we select the same number of phases as in the 2D case, which corresponds to $16^4 = 65\,536$ possible combinations. This restricts the number of possible gait patterns but still allows us to examine whether the 2D conclusions also occur for the 3D case. The result is illustrated in Fig. 46, and some example motions are shown in Fig. 49.

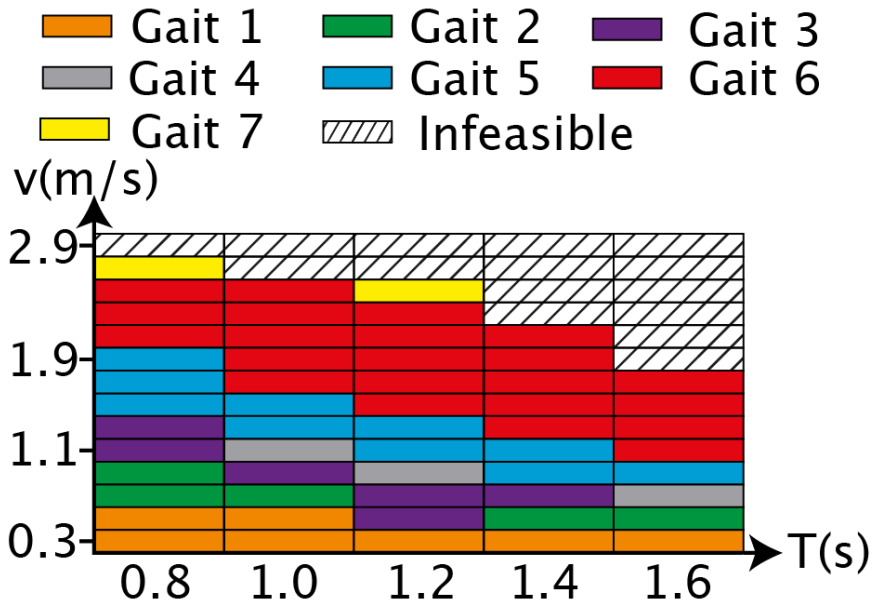


Figure 46: Optimal gait pattern map for the 3D quadruped.

In this 3D case, we observe a structured gait pattern map too. For low speeds, the model favors a series of gait patterns that avoid flying phases (gaits 1 to 5 as shown in Fig. 47). In higher speed regimes, the model transits to gait patterns with noticeable flying phases such as running trot and pace (gaits 6 and 7). Note that the number of gait patterns we obtained is even larger now, demonstrating further the need to automate the gait pattern selection aspect.

Next, we focus on the properties of state trajectories within individual gait pattern regions. We noticed that, similar to the case of the 2D half-cheetah model, the shape of

the state-space trajectories remains the same within a gait region, and only the scale of the trajectories changes, e.g., the state-space trajectories of the running-trot gait in Fig. 48. To test the effectiveness of the warm-staring strategy in the 3D case, we randomly pick 10 task descriptors and solve them with the same two TO methods: one with pre-specified gait pattern only and one with both gait pattern and nominal state initialization.

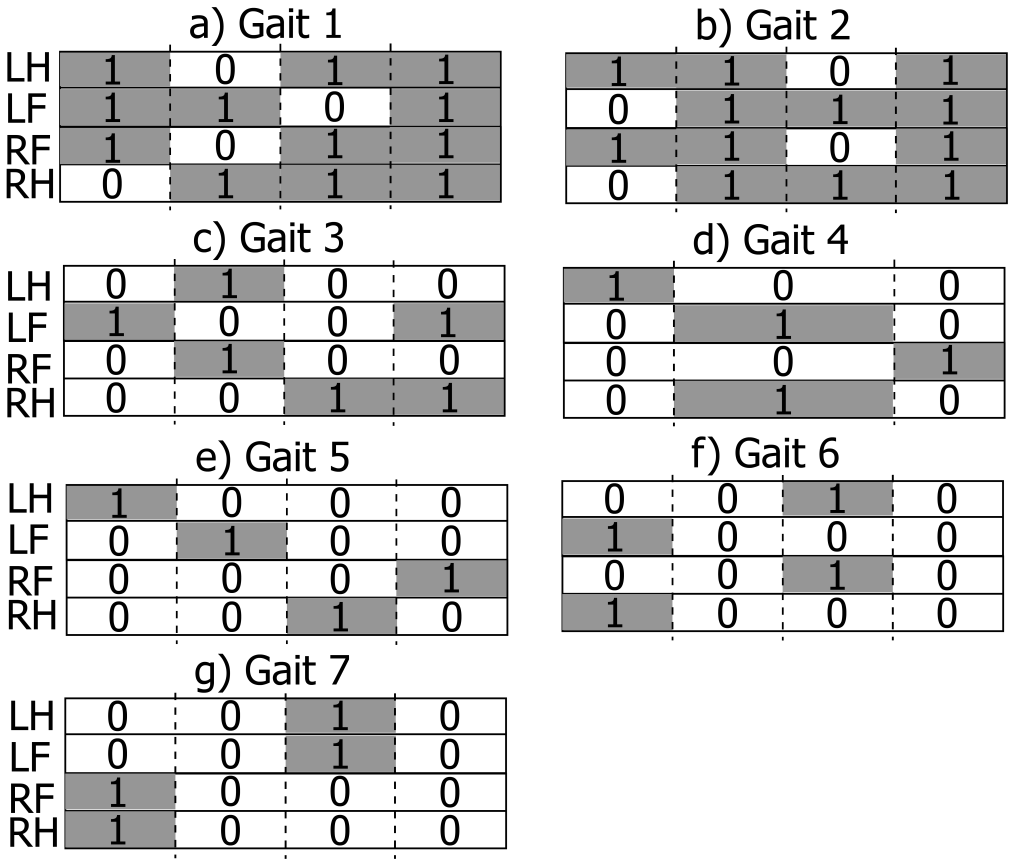


Figure 47: Discovered gait patterns for the 3D quadruped with four phases.

We report the results of the comparison in Table 8. The optimization problems now have 1948 decision variables and 5008 constraints, and become significantly more challenging to solve. Still, pre-specifying gait patterns and initializing with nominal state trajectories lead to a large computational improvements, without sacrificing the number of convergence and the consistency.

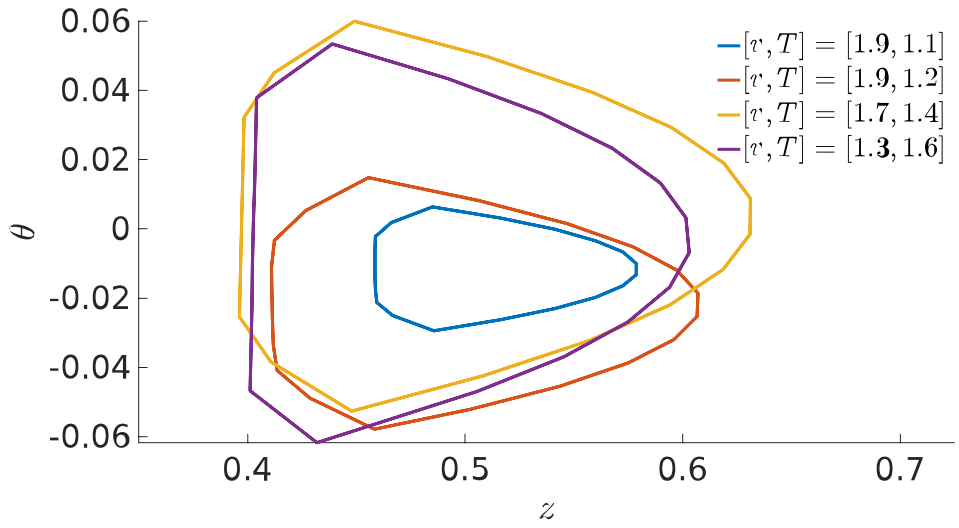


Figure 48: Examples of the state-space trajectory for Gait 6 (Running-Trot). θ is the pitch angle of the base in radius, and z is the CoM height of the base in meters.

Table 8: Computation performance between the MINLP and the two TO methods for computing 3D quadruped locomotion plans with pre-specified optimal gait pattern, with and without nominal state initialization. We test 10 randomly sampled task specifications for the 3D quadruped case.

Method	Computation time	Successful Convergence	Consistency
MINLP	5+ hours	10/10	N/A
TO (Gait pattern only)	19.136±10.23 seconds	7/10	5/10
TO (Gait pattern & warm-state)	**0.3159±0.05** seconds	10/10	10/10

Figure 49: Snapshots of different gait pattern discovered for 3D quadruped robot ANYmal: three-beat walking (Gait 1), walking trot (Gait 2), four-beat walking (Gait 5) and running trot (Gait 6). Red arrows represent contact force vectors.

9.4.4.1 *Real-World Robot Experiment*

To validate the dynamic feasibility of the trajectories computed by our method, we adapt the limits of contact force and foot velocity to be close to the capabilities of the ANYmal robot, and we demonstrate a real-world trial (Fig. 50) of a motion computed by our MINLP approach. We use the whole-body inverse dynamics controller developed in [114] to track the planned motion.

9.5 CONCLUSION

In this chapter, we presented a framework that can automate gait pattern selection for legged robots. The core idea of our approach is to build a map from task specifications to gait pattern selection for a given environment and performance objective. This map can allow us to quickly select gait patterns in a systematical fashion, without requiring heuristics or input from a human operator. Following this idea, we established gait pattern selection maps for a 2D half-cheetah model and the 3D quadruped robot ANYmal. We show that these maps typically consist of several contiguous regions for which a particular gait pattern is optimal, and we can use supervised learning to encode the structure of the map efficiently. Furthermore, we also found that the trajectories within each gait region are qualitatively similar. This suggests that these trajectories are in the same basin of attraction. We empirically find that we can use the mean of these trajectories to initialize our optimization-based planning algorithm. By utilizing the gait pattern selection maps together with the obtained initial seeds, we demonstrate a significant improvement in the computation time for our trajectory optimization method, e.g., reducing the computation time from hours or minutes to 0.1-0.3 seconds.

To extend our work, we believe the investigation of the following two aspects is necessary. First, we suggest to focus on addressing the scalability of high-dimensional models (for example, consider the robot model at a whole-body level), a large number of phases, and complex environments [26, 87]. These problems stem from either the large number of variables that increase the complexity of the problem, or from the non-convexity of the dynamics model.

Second, a more systematic study is required regarding the structural pattern of the optimal gait pattern map, as well as the trajectory properties within the regions of the map. In this work, we exploited the structure of the map and the similarity of state trajectories to warm-start TO methods. We believe these findings can be generalized for different tasks (for example, acceleration and deceleration), performance object-

ives, and environments. The computational improvements that we enjoy using these properties can allow us to achieve online motion re-planning in real-world scenarios; the human operator will specify speed, while the gait pattern and state initialization will be automatically selected to bootstrap the computation.

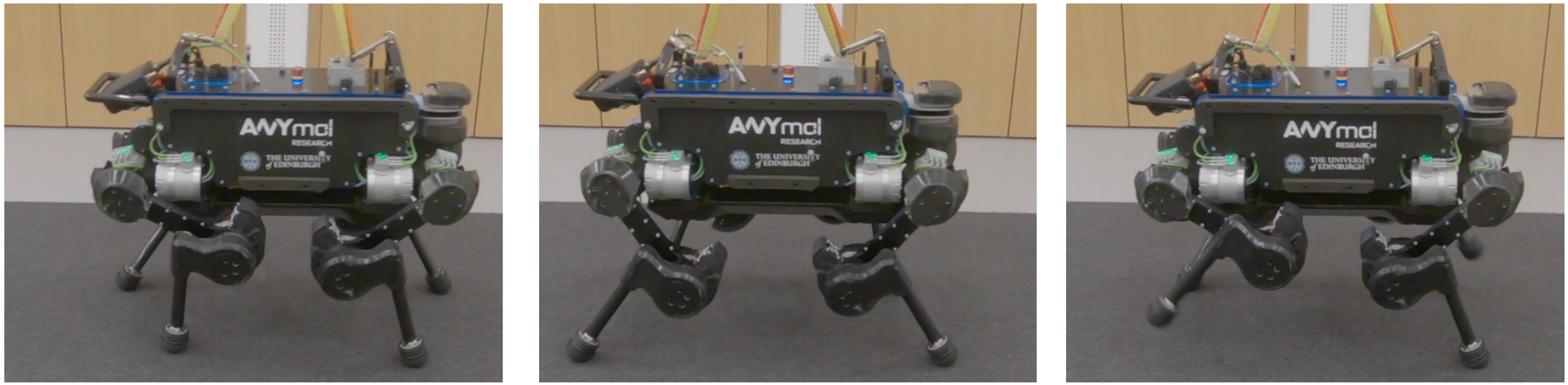


Figure 50: Snapshots of an experimental trial on the ANYmal robot. For the performance objective given in Section 9.4.4, our proposed framework discovers that the walking-trot gait is preferred for the locomotion task with a forward speed of $0.3m/s$ and cycle duration of $0.8s$ (Video: https://youtu.be/BHfPymX6Hhs).

10

CONCLUSIONS AND FUTURE WORK

10.1 THESIS SUMMARY

In this thesis, we have presented our explorations towards online Receding Horizon Planning (RHP) of multi-contact motions. Generally speaking, computing multi-contact motion plans is usually time-consuming, as this problem is combinatorial, high-dimensional, and non-convex. As discussed in Section 1.2, these computational challenges typically stem from the joint resolution of the following two sub-problems:

- Sub-problem 1: Gait pattern selection, which refers to the issue of determining the sequence in which the limbs break and make contact with the ground. This gait pattern selection issue is the primary source of combinatorial complexity.

- Sub-problem 2: Contact and motion planning, which refers to the problem of finding a feasible state trajectory along with a contact plan (for example, locations and timings) that is consistent with the chosen gait pattern. This sub-problem typically features high-dimensionality and non-convexity due to the consideration of complex non-linear dynamics constraints.

To facilitate online RHP of multi-contact motions, in this thesis, we have explored several methods to address these two sub-problems efficiently. More specifically, in Chapter 4 to Chapter 8, we focused on the case of achieving online RHP of contact and motion plans (sub-problem 2). In this case, the gait pattern is pre-specified as a priori (an ad-hoc solution of sub-problem 1). To enable online computation, the core idea of our approaches is to reduce the computation burden introduced by the value function approximation that is required for guiding the RHP framework. Then, in Chapter 9, we focused on dealing with the combinatorial complexity that arises from the gait pattern selection issue. To give more detail, we propose to leverage offline computations to build a map from the task conditions to the gait pattern selection. By querying this map, we can quickly decide the gait pattern without resolving the combinatorial complexity.

In the following sections, we summarize the key ideas and the results presented in Chapter 4 to Chapter 9.

10.1.1 *Summary of Online Receding Horizon Planning (RHP) of Contact and Motion Plans*

When planning contact and motion plans in a receding horizon fashion, it is important to consider a value function model to guide the execution horizon (motion being executed) toward a state that is beneficial for the future. To model this value function, traditional RHP approaches often approximate it by computing trajectories in a prediction horizon (not executed) that foresees the future. Nevertheless, approximating the value function with the prediction horizon typically results in expensive computations, especially when computing contact and motion plans where complex nonlinear dynamics constraints need to be considered.

To accelerate the computation speed, we explore the possibility to find computationally efficient approximations of the value function. This leads to:

- *Receding Horizon Planning with Multiple Levels of Model Fidelity*, where we approximate the value function by computing the prediction horizon with convex relaxed models. In this thesis, we explore and compare three multi-fidelity RHP candidates with different convex relaxations employed in the prediction horizon.

- *Locally-Guided Receding Horizon Planning*, where we approximate the value function with a learned oracle that can predict local objectives as intermediate goals for a given task. These local objectives are then used to construct local value functions to guide a short-horizon RHP.

We evaluate the computation performance of multi-fidelity RHP and locally-guided RHP in the context of planning centroidal trajectories of the humanoid robot Talos [90] walking on uneven terrain, and we use a whole-body inverse dynamics controller [27] to validate the dynamic feasibility of the planned trajectories in simulation. Our experiment result shows that both multi-fidelity RHP and locally-guided RHP can achieve online RHP of contact and motion plans. Furthermore, due to the shortened planning horizon, the locally-guided RHP achieves the best online convergence rate. In addition, we also validated locally-guided RHP with real-world robot experiments, where we demonstrated online receding horizon planning of contact and motion plans on our humanoid robot Talos [90] in dynamically changing environments.

10.1.2 *Summary of Automatic Gait Pattern Selection*

To address the combinatorial complexity arising from the gait pattern selection issue, we propose the idea of establishing a map that can tell the gait pattern selection

based on the task specifications and the environment model. To generate the map, we employ Mixed-Integer Non-Linear Programming (MINLP) to offline compute a dataset of optimal gait patterns for a set of task specifications and environments. Then, we use a neural network to encode the map. Here the term "optimal" refers to the locally optimal solutions discovered by re-starting the computation with different initial guesses.

Following the idea, we construct optimal gait pattern selection maps for a 2D half-cheetah model and the 3D quadruped robot ANYmal. We find that these maps are composed of several contiguous regions for which a particular gait pattern is optimal. Moreover, we also notice that the optimal trajectories within each gait region are qualitatively similar. Based on this property, we find that we can use the mean of these trajectories to warm-start our TO based planning algorithm. Lastly, we also performed an experimental trial on the ANYmal robot to validate the motion planned by our method.

10.2 LIMITATIONS AND FUTURE DIRECTIONS

In this section, we outline the limitations of our work and discuss the potential directions of future research.

10.2.1 *Limitations of our Online Receding Horizon Planning (RHP) Methods for Computing Contact and Motion Plans*

This section summarizes the limitations of our approaches for achieving online RHP of contact and motion plans. Furthermore, we also highlight future research directions that may improve their performance.

10.2.1.1 *Improving the Accuracy of Value Function Approximation and Complementary Methods to Ensure Safe Operation*

In Chapter 7, we have shown that owing to the simplifications introduced into the value function approximation, both multi-fidelity RHP and locally-guided RHP can achieve fast computation of contact and motion plans. Nevertheless, these simplifications can also affect the accuracy of the value function, which can sometimes cause convergence failures, i.e., the RHP frameworks can be directed into an ill-posed state, from which the TO can fail to find a feasible solution.

To deal with the issue of convergence failures, we believe it is important to improve the accuracy of the value function approximation employed in our RHP methods,

while in the meantime maintaining their efficient computation. For instance, for multi-fidelity RHP, instead of using convex relaxed models to plan the prediction horizon which may violate the system dynamics constraint, we can also consider using convex inner approximation models to compute the prediction horizon [35]. We speculate that the convex inner approximation models can provide more conservative and accurate value function approximations, which may alleviate the convergence failure issue. Alternatively, we can try to capture the complex system dynamics with a learned convex parametric model. Ideally, by planning the prediction horizon with this learned model, we could improve the accuracy of the value function approximation while remaining efficient computation. For the locally-guided RHP, the quality of the value function approximation heavily depends on the prediction accuracy of the oracle. In this thesis, we have shown that we can improve the prediction accuracy of the oracle by augmenting the data set with the data points that demonstrate corrective actions from the states that cause convergence failures. However, we still found it hard to achieve 100% prediction accuracy, showing that further investigation on the data augmentation technique is necessary.

On the other hand, we should note that introducing simplifications into the value function approximation will inevitably affect its accuracy. This trade-off implies that, although there are still opportunities to improve the accuracy of our value function approximation approach, we may find it difficult to achieve 100% convergence rate of RHP. As a result, we may need to consider complementary methods to protect the robot from convergence failures. To give more detail, the main issue arises from convergence failures is that the robot may be led to an unstable state and start losing balance. In that case, we can use the fall protection method to try to bring the robot to a halt if we detect the robot is failing [29]. This can help us to minimize the damage of the falling to the robot. Also, if the robot already falls on the ground, we can use the fall recovery method [115] to bring the robot back to a nominal posture. We believe fall protection and fall recovery techniques are also important to facilitate safe operation of legged robots.

10.2.1.2 *Towards More Reactive Computation*

In this thesis, we test both multi-fidelity RHP and locally-guided RHP in an online setting, where in each planning cycle, we require the planner to compute the motion plan for the next step before finishing the execution of the current step. Such an online setting can allow the robot to achieve reliable locomotion in response to the environment changes, and we show that both multi-fidelity RHP and locally-guided RHP can achieve online computation in this setting.

Nevertheless, when deploying legged robots in the real world, the robot can meet more instantaneous perturbations, i.e., state deviations caused by external forces applied on the robot, footstep deviations due to imperfect control or foot slippage. To cope with these abrupt changes, legged robots necessarily require the capability to update their motions in a more reactive fashion. To achieve this, we need to further improve the computation efficiency of our methods.

To be more specific, for multi-fidelity RHP, the first avenue we should explore is to find more simplified models to plan the prediction horizon. However, as mentioned in Section 10.2.1.1, these further simplifications should not affect the accuracy of the value function approximation. Alternatively, we can also improve the computation speed of multi-fidelity RHP by exploiting the structure of the RHP formulation. More specifically, given the prediction horizon is fully convex, we can split the planning of the execution horizon (non-convex part) and the prediction horizon (convex part) into two sub-problems, and then we solve these two sub-problems alternatively until convergence [10]. To further improve the computation speed of the locally-guided RHP, we should focus on achieving more efficient computation of the execution horizon, as the computation burden of the prediction horizon is removed by a learned value function model. To achieve this goal, we can reduce the dimensionality of the execution horizon by modeling the trajectories with parametric curves, i.e., bezier curve [35]. Alternatively, we can also explore the possibility to warm-start the computation of the execution horizon [58].

10.2.1.3 *Extending to Whole-body Motion Planning/Control*

In this thesis, we test both multi-fidelity RHP and locally-guided RHP in the case of planning centroidal trajectories of the humanoid robot Talos walking on uneven terrain. As mentioned in Section 2.1.2, the centroidal dynamics model features lower dimensionality, which can allow us to have more tractable computation. Nevertheless, the centroidal dynamics model introduces approximations on robot kinematics and assumes that the robot always has sufficient joint torques to achieve the desired centroidal momenta. These assumptions can lead to failures in achieving a corresponding whole-body motion. Given this issue, we believe it is worthwhile to extend our studies to the case of planning whole-body motion plans. Additionally, considering the whole-body dynamics model also allows us to fully exploit the dynamics of each individual link, which can enable the generation of more dynamics motions, i.e., jumping and back-flipping [62, 74].

10.2.2 *Limitations of our Automatic Gait Pattern Selection Approach*

To address the combinatorial complexity of the multi-contact motion planning problem, in this thesis, we present a learning-based framework to build maps that can select the gait pattern based on the task specifications and the environment model. To generate the data set, we rely on a MINLP approach to offline search for the optimal gait patterns of different task specifications and environment models. Nevertheless, as shown in Chapter 9, this MINLP approach does not scale with a large number of integer variables. For example, when we switch from the half-cheetah model to the 3D quadruped, the number of possible gait patterns increases from 256 to 65536 (although we assume the motion always contains 4 contact phases), and we witness an exponential growth of the computation time (from 90s to 5 hours). As we can imagine, if we further increase the combinatorial complexity of the problem by considering more complex scenarios, the MINLP problem may even become intractable for offline computation, i.e., increasing the number of contact phases and the number of limbs (hexapod), taking into account more complex environments composed of disjoint contact surfaces [2, 26, 80, 87]. To alleviate this scalability issue, we can develop pre-processing methods to filter out the gait patterns that are kinematically infeasible. Alternatively, we can also develop heuristics to improve the search efficiency of the branch and bound algorithm [50].

10.2.3 *Combining Our Methods for Gait Pattern Selection and Contact and Motion Planning*

In an ideal case, the multi-contact motion planning problem should be solved by jointly optimizing the discrete decisions—gait pattern selection and/or contact surface selection, along with the continuous decisions—computing a feasible contact and motion plan. Although in theory, Mixed-Integer Programming (MIP) is the most general way to address such an optimization problem that involves both discrete and continuous variables, we have shown that MIP methods suffer from long computation time. The computational burden typically results from the Branch-and-Bound (B&B) algorithm which deals with the combinatorial aspect of the problem: a large number of complex inner optimization problems need to be addressed to evaluate the feasibility and optimality of the motions characterized by different gait patterns. Although state-of-the-art legged locomotion planning algorithms have enabled the efficient computation of these inner problems and can significantly improve the computation time of MIP [80], this approach still finds it difficult to achieve online computation when there exists a large number of integer variables need to be addressed.

Alternatively, to achieve efficient computation, we can also address the multi-contact motion planning problem in a decoupled fashion. For instance, we can firstly decide the discrete variables (gait pattern and/or contact surface selection), and then compute the continuous variables (contact and motion plan). In this thesis, we have presented novel methods that can efficiently address the discrete part and the continuous part of the multi-contact planning problem respectively. For example, we propose to offline build a map for selecting the gait pattern, and we also achieve online RHP of contact and motion plans over uneven terrain (with fixed gait pattern) via efficient value function approximation methods. In the future, we can explore the combination of these methods to build a complete computation pipeline to achieve online multi-contact motion planning in challenging environments. Nevertheless, as mentioned in Section 10.2.2, a core challenge of achieving this pipeline is that the offline computation involved in building the gait pattern selection map can become prohibitive when we consider more complex scenarios, i.e., searching for a large number of integer variables due to the consideration of more contact phases or limbs (using hands to assist stair climbing), or taking into account contact surface selection [26, 87]. Probably, instead of relying on offline computation to build the map, we could explore the possibility of learning the map from human/animal demonstrations [48, 117]. We believe further research in building this decoupled computation pipeline is a promising research direction.

10.3 EPILOGUE

In this thesis, we explored a couple of novel methods for achieving online RHP of multi-contact motions. The core idea of our methods is to simplify the computation complexity of the original planning problem by introducing model simplifications and domain knowledge learned from past experiences. The impact of our methods lies in their fast computation speed, which can enable legged robots to online adapt their motions in response to unexpected changes, i.e., dynamic changes of the environment or task changes from a human operator. Such capability is essential for achieving reliable operation in uncertain environments. Looking into the future, we can further improve the performance of our methods in the following aspects, i.e., improving the convergence rate, achieving more reactive computation, scaling to more complex robot models and scenarios, etc. We envision that in the near future, legged robots can become ready for real-world applications, and undertake a wide range of tasks that are tedious and dangerous for humans.

BIBLIOGRAPHY

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard et al. 'TensorFlow: a system for Large-Scale machine learning'. In: *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 2016, pp. 265–283 (cit. on p. 59).

[2] Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G Caldwell, José Cappelletto, Juan C Grieco, Gerardo Fernández-López and Claudio Semini. 'Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization'. In: *IEEE Robotics and Automation Letters (RA-L)* 3.3 (2018), pp. 2531–2538 (cit. on pp. 17, 20, 83, 105, 123, 136).

[3] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings and Moritz Diehl. 'CasADi – A software framework for nonlinear optimization and optimal control'. In: *Mathematical Programming Computation* 11.1 (July 2018), pp. 1–36 (cit. on pp. 69, 90, 115).

[4] Manuel Armada, Pablo González De Santos, María A Jiménez and Manuel Prieto. 'Application of CLAWAR machines'. In: *International Journal of Robotics Research (IJRR)* 22.3-4 (2003), pp. 251–264 (cit. on p. 2).

[5] Christopher G Atkeson, Benzun P Wisely Babu, Nandan Banerjee, Dmitry Berenson, Christoper P Bove, Xiongyi Cui, Mathew DeDonato, Ruixiang Du, Siyuan Feng, Perry Franklin et al. 'No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge'. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2015, pp. 623–630 (cit. on p. 2).

[6] C Dario Bellicoso, Fabian Jenelten, Christian Gehring and Marco Hutter. 'Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots'. In: *IEEE Robotics and Automation Letters (RA-L)* 3.3 (2018), pp. 2261–2268 (cit. on pp. 12, 13, 15, 105).

[7] Richard Bellman. 'Dynamic programming'. In: *Science* 153.3731 (1966), pp. 34–37 (cit. on pp. 2, 29–32).

[8] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010 (cit. on pp. 2, 15, 30).

[9] Christopher M. Bishop. *Mixture Density Networks*. Tech. rep. Aston University, 1994 (cit. on p. 84).

[10] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein et al. 'Distributed optimization and statistical learning via the alternating direction method of multipliers'. In: *Foundations and Trends in Machine learning* 3.1 (2011), pp. 1–122 (cit. on p. 135).

[11] Timothy Bretl. 'Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem'. In: *International Journal of Robotics Research (IJRR)* 25.4 (2006), pp. 317–342 (cit. on pp. 2, 5, 12).

[12] Richard H. Byrd, Jorge Nocedal and Richard A. Waltz. 'Knitro: An Integrated Package for Nonlinear Optimization'. In: *Large-Scale Nonlinear Optimization*. Springer US, 2006, pp. 35–59 (cit. on pp. 69, 90, 115).

[13] Jan Carius, René Ranftl, Vladlen Koltun and Marco Hutter. 'Trajectory optimization with implicit hard contacts'. In: *IEEE Robotics and Automation Letters (RA-L)* 3.4 (2018), pp. 3316–3323 (cit. on p. 19).

[14] Stéphane Caron and Abderrahmane Kheddar. 'Multi-contact walking pattern generation based on model preview control of 3d com accelerations'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2016, pp. 550–557 (cit. on p. 16).

[15] Stéphane Caron and Quang-Cuong Pham. 'When to make a step? tackling the timing problem in multi-contact locomotion by topp-mpc'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2017, pp. 522–528 (cit. on p. 16).

[16] J. Carpentier and N. Mansard. 'Multicontact Locomotion of Legged Robots'. In: *IEEE Transactions on Robotics (T-RO)* 34.6 (2018) (cit. on pp. 5, 16, 105).

[17] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse and Nicolas Mansard. 'A versatile and efficient pattern generator for generalized legged locomotion'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 3555–3561 (cit. on p. 16).

[18] Jason Chemin, Pierre Fernbach, Daeun Song, Guilhem Saurel, Nicolas Mansard and Steve Tonneau. 'Learning to steer a locomotion contact planner'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 4430–4437 (cit. on p. 15).

[19] Alessandro Chiuso and Gianluigi Pillonetto. 'System identification: A machine learning perspective'. In: *Annual Review of Control, Robotics, and Autonomous Systems* 2 (2019), pp. 281–304 (cit. on p. 104).

[20] Jens Clausen. 'Branch and bound algorithms-principles and examples'. In: *Department of Computer Science, University of Copenhagen* (1999) (cit. on p. 20).

[21] Thomas Cover and Peter Hart. 'Nearest neighbor pattern classification'. In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27 (cit. on p. 61).

[22] Hongkai Dai and Russ Tedrake. 'Planning robust walking motion on uneven terrain via convex optimization'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2016, pp. 579–586 (cit. on p. 17).

[23] Hongkai Dai, Andrés Valenzuela and Russ Tedrake. 'Whole-body motion planning with centroidal dynamics and full kinematics'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2014 (cit. on p. 16).

[24] Ewen Dantec, Rohan Budhiraja, Adria Roig, Teguh Lembono, Guilhem Saurel, Olivier Stasse, Pierre Fernbach, Steve Tonneau, Sethu Vijayakumar, Sylvain Calinon, Michel Taïx and Nicolas Mansard. 'Whole Body Model Predictive Control with a Memory of Motion: Experiments on a Torque-Controlled Talos'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021 (cit. on pp. 2, 18, 30).

[25] Robin Deits, Twan Koolen and Russ Tedrake. 'LVIS: Learning from value function intervals for contact-aware robot controllers'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7762–7768 (cit. on pp. 18, 19, 31, 36, 57).

[26] Robin Deits and Russ Tedrake. 'Footstep planning on uneven terrain with mixed-integer convex optimization'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2014, pp. 279–286 (cit. on pp. 20, 85, 127, 136, 137).

[27] Andrea Del Prete and Nicolas Mansard. 'Robustness to joint-torque-tracking errors in task-space inverse dynamics'. In: *IEEE Transactions on Robotics (T-RO)* 32.5 (2016), pp. 1091–1105 (cit. on pp. 7, 69, 132).

[28] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt and Sangbae Kim. 'Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9 (cit. on pp. 2, 30).

[29] Wenpeng Ding, Xuechao Chen, Zhangguo Yu, Libo Meng, Marco Ceccarelli and Qiang Huang. 'Fall protection of humanoids inspired by human fall motion'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2018, pp. 827–833 (cit. on p. 134).

[30] Johannes Englsberger, Christian Ott and Alin Albu-Schäffer. 'Three-dimensional bipedal walking control based on divergent component of motion'. In: *IEEE Transactions on Robotics (T-RO)* 31.2 (2015), pp. 355–368 (cit. on p. 16).

[31] Tom Erez and Emanuel Todorov. 'Trajectory optimization for domains with contacts using inverse dynamics'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 4914–4919 (cit. on p. 16).

[32] Adrien Escande, Abderrahmane Kheddar, Sylvain Miossec and Sylvain Garsault. 'Planning support contact-points for acyclic motions and experiments on HRP-2'. In: *Experimental Robotics*. Springer. 2009, pp. 293–302 (cit. on p. 12).

[33] Péter Fankhauser, Marko Bjelonic, C Dario Bellicoso, Takahiro Miki and Marco Hutter. 'Robust rough-terrain locomotion with a quadrupedal robot'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5761–5768 (cit. on pp. 12, 13, 15).

[34] Siyuan Feng, X Xinjilefu, Christopher G Atkeson and Joohyung Kim. 'Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals'. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2015, pp. 1028–1035 (cit. on p. 2).

[35] Pierre Fernbach, Steve Tonneau, Olivier Stasse, Justin Carpentier and Michel Taïx. 'C-CROC: Continuous and Convex Resolution of Centroidal Dynamic Trajectories for Legged Robots in Multicontact Scenarios'. In: *IEEE Transactions on Robotics (T-RO)* 36.3 (2020), pp. 676–691 (cit. on pp. 15, 16, 38, 42, 84, 134, 135).

[36] Mathieu Geisert, Thomas Yates, Asil Orgen, Pierre Fernbach and Ioannis Havoutis. 'Contact planning for the anymal quadruped robot using an acyclic reachability-based planner'. In: *Annual Conference Towards Autonomous Robotic Systems*. Springer. 2019, pp. 275–287 (cit. on p. 15).

[37] Markus Giftthaler, Michael Neunert, Markus Stäuble, Jonas Buchli and Moritz Diehl. 'A family of iterative gauss-newton shooting methods for nonlinear optimal control'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9 (cit. on p. 16).

[38] Michael X Grey, Aaron D Ames and C Karen Liu. 'Footstep and motion planning in semi-unstructured environments using randomized possibility graphs'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 4747–4753 (cit. on p. 12).

[39] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2023. URL: https://www.gurobi.com (cit. on p. 20).

[40] A. Herzog, S. Schaal and L. Righetti. 'Structured contact force optimization for kino-dynamic motion generation'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2703–2710 (cit. on p. 16).

[41] Alexander Herzog, Nicholas Rotella, Stefan Schaal and Ludovic Righetti. 'Trajectory generation for multi-contact momentum-control'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2015 (cit. on p. 16).

[42] Kurt Hornik, Maxwell Stinchcombe and Halbert White. 'Multilayer feedforward networks are universal approximators'. In: *Neural Networks* 2.5 (1989), pp. 359–366 (cit. on p. 62).

[43] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch et al. 'Anymal-a highly mobile and dynamic quadrupedal robot'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 38–44 (cit. on p. 106).

[44] Simon J Julier and Jeffrey K Uhlmann. 'Unscented filtering and nonlinear estimation'. In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422 (cit. on p. 99).

[45] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi and Hirohisa Hirukawa. 'Biped walking pattern generation by using preview control of zero-moment point'. In: *IEEE International*

Conference on Robotics and Automation (ICRA). Vol. 2. 2003, pp. 1620–1626 (cit. on p. 16).

[46] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry and Stefan Schaal. 'Fast, robust quadruped locomotion over challenging terrain'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 2665–2670 (cit. on p. 105).

[47] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry and Stefan Schaal. 'Learning, planning, and control for quadruped locomotion over challenging terrain'. In: *International Journal of Robotics Research (IJRR)* 30.2 (2011), pp. 236–258 (cit. on pp. 12, 13).

[48] Dongho Kang, Flavio De Vincenti, Naomi C Adami and Stelian Coros. 'Animal Motions on Legged Robots Using Nonlinear Model Predictive Control'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 11955–11962 (cit. on p. 137).

[49] Matthew Kelly. 'An introduction to trajectory optimization: How to do your own direct collocation'. In: *SIAM Review* 59.4 (2017), pp. 849–904 (cit. on p. 15).

[50] Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser and Bistra Dilkina. 'Learning to branch in mixed integer programming'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016 (cit. on p. 136).

[51] Kai Henning Koch, Katja Mombaur and Philippe Soueres. 'Optimization-based walking generation for humanoid robot'. In: *IFAC Proceedings Volumes* 45.22 (2012), pp. 498–504 (cit. on p. 16).

[52] J Zico Kolter, Mike P Rodgers and Andrew Y Ng. 'A control architecture for quadruped locomotion over rough terrain'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2008, pp. 811–818 (cit. on pp. 12–14).

[53] Jens Koschorreck and Katja Mombaur. 'Modeling and optimal control of human platform diving with somersaults and twists'. In: *Optimization and Engineering* 13.1 (2012), pp. 29–56 (cit. on p. 16).

[54] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt and Christopher Orlowski. 'The DARPA robotics challenge finals: Results and perspectives'. In: *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*. Springer, 2018, pp. 1–26 (cit. on p. 2).

[55] LAAS-CNRS. *Gepetto-Viewer*. 2023. URL: https://github.com/Gepetto/gepetto-viewer (cit. on p. 11).

[56] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan and Ken Goldberg. 'Dart: Noise injection for robust imitation learning'. In: *Conference on Robot Learning*. PMLR. 2017, pp. 143–156 (cit. on p. 19).

[57] T. S. Lembono, C. Mastalli, P. Fernbach, N. Mansard and S. Calinon. 'Learning How to Walk: Warm-starting Optimal Control Solver with Memory of Motion'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1357–1363 (cit. on p. 18).

[58] Teguh Santoso Lembono. 'Memory of Motion for Initializing Optimization in Robotics'. PhD thesis. EPFL, 2022 (cit. on p. 135).

[59] He Li, Robert J. Frei and Patrick M. Wensing. 'Model Hierarchy Predictive Control of Robotic Systems'. In: *IEEE Robotics and Automation Letters (RA-L)* 6.2 (2021), pp. 3373–3380 (cit. on pp. 17, 52).

[60] Yu-Chi Lin, Brahayam Ponton, Ludovic Righetti and Dmitry Berenson. 'Efficient humanoid contact planning using learned centroidal dynamics prediction'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5280–5286 (cit. on p. 18).

[61] Ian Manchester, Uwe Mettin, Fumiya Iida and Russ Tedrake. 'Stable dynamic walking over rough terrain: Theory and experiment'. In: *International Symposium on Robotics Research (ISRR)*. Springer-Verlag. 2009, pp. 1–16 (cit. on p. 16).

[62] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar and N. Mansard. 'Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020 (cit. on pp. 2, 5, 15, 16, 30, 104, 105, 135).

[63] Carlos Mastalli, Michele Focchi, Ioannis Havoutis, Andreea Radulescu, Sylvain Calinon, Jonas Buchli, Darwin G Caldwell and Claudio Semini. 'Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1096–1103 (cit. on p. 105).

[64] Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G. Caldwell and Claudio Semini. 'Hierarchical planning of dynamic movements without sched-

uled contact sequences'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016 (cit. on p. 19).

[65] Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G. Caldwell and Claudio Semini. 'Motion Planning for Quadrupedal Locomotion: Coupled Planning, Terrain Mapping and Whole-Body Control'. In: *IEEE Transactions on Robotics (T-RO)* (2020) (cit. on p. 107).

[66] Carlos Mastalli, Wolfgang Merkt, Josep Marti-Saumell, Henrique Ferrolho, Joan Solà, Nicolas Mansard and Sethu Vijayakumar. 'A feasibility-driven approach to control-limited DDP'. In: *Autonomous Robots* (2022), pp. 1–21 (cit. on p. 16).

[67] Avadesh Meduri, Paarth Shah, Julian Viereck, Majid Khadiv, Ioannis Havoutis and Ludovic Righetti. 'Biconmp: A nonlinear model predictive control framework for whole body motion planning'. In: *IEEE Transactions on Robotics (T-RO)* (2023) (cit. on p. 16).

[68] Oliwier Melon, Romeo Orsolino, David Surovik, Mathieu Geisert, Ioannis Havoutis and Maurice Fallon. 'Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9805–9811 (cit. on pp. 2, 18, 30).

[69] T. Moore and D. Stouch. 'A Generalized Extended Kalman Filter Implementation for the Robot Operating System'. In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014 (cit. on p. 98).

[70] Igor Mordatch, Emanuel Todorov and Zoran Popović. 'Discovery of complex behaviors through contact-invariant optimization'. In: *ACM Transactions on Graphics (TOG)* 31.4 (2012) (cit. on p. 19).

[71] Kourosh Naderi, Joose Rajamäki and Perttu Hämäläinen. 'Discovering and Synthesizing Humanoid Climbing Movements'. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017) (cit. on p. 106).

[72] Michael Neunert, Farbod Farshidian, Alexander W Winkler and Jonas Buchli. 'Trajectory optimization through contacts and automatic gait discovery for quadrupeds'. In: *IEEE Robotics and Automation Letters (RA-L)* 2.3 (2017), pp. 1502–1509 (cit. on p. 19).

[73] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter and Jonas Buchli. 'Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds'. In: *IEEE Robotics and Automation Letters (RA-L)* 3.3 (2018), pp. 1458–1465 (cit. on pp. 2, 30, 107).

[74] Quan Nguyen, Matthew J Powell, Benjamin Katz, Jared Di Carlo and Sangbae Kim. 'Optimized jumping on the mit cheetah 3 robot'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7448–7454 (cit. on p. 135).

[75] David E. Orin, Ambarish Goswami and Sung Hee Lee. 'Centroidal dynamics of a humanoid robot'. In: *Autonomous Robots* 35.2-3 (2013), pp. 161–176 (cit. on pp. 5, 16, 42).

[76] Amit Parag, Sébastien Kleff, Léo Saci, Nicolas Mansard and Olivier Stasse. 'Value learning from trajectory optimization and Sobolev descent: A step toward reinforcement learning with superlinear convergence properties'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2022 (cit. on p. 18).

[77] Hae-Won Park, Patrick M Wensing and Sangbae Kim. 'Online planning for autonomous running jumps over obstacles in high-speed quadrupeds'. In: *Robotics: Science and Systems*. 2015 (cit. on pp. 2, 29, 30, 105).

[78] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 114).

[79] Brahayam Ponton, Alexander Herzog, Stefan Schaal and Ludovic Righetti. 'A convex model of humanoid momentum dynamics for multi-contact motion generation'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2016, pp. 842–849 (cit. on p. 20).

[80] Brahayam Ponton, Majid Khadiv, Avadesh Meduri and Ludovic Righetti. 'Efficient multicontact pattern generation with sequential convex approximations of the centroidal dynamics'. In: *IEEE Transactions on Robotics (T-RO)* (2021) (cit. on pp. 17, 45, 49, 50, 67, 105, 123, 136).

[81] Michael Posa, Cecilia Cantu and Russ Tedrake. 'A direct method for trajectory optimization of rigid bodies through contact'. In: *International Journal of Robotics Research (IJRR)* 33.1 (2014), pp. 69–81 (cit. on pp. 2, 19, 29, 84, 114).

[82] John R Rebula, Peter D Neuhaus, Brian V Bonnlander, Matthew J Johnson and Jerry E Pratt. 'A controller for the littledog quadruped walking on rough terrain'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 1467–1473 (cit. on pp. 12–14).

[83] C David Remy. 'Optimal exploitation of natural dynamics in legged locomotion'. PhD thesis. ETH Zurich, 2011 (cit. on pp. 4, 105).

[84] Stéphane Ross, Geoffrey Gordon and Drew Bagnell. 'A reduction of imitation learning and structured prediction to no-regret online learning'. In: *International Conference on Artificial Intelligence and Statistics*. JMLR. 2011, pp. 627–635 (cit. on pp. 19, 63, 65).

[85] Gerrit Schultz and Katja Mombaur. 'Modeling and optimal control of human-like running'. In: *IEEE/ASME Transactions on Mechatronics* 15.5 (2009), pp. 783–792 (cit. on p. 16).

[86] Matthias Seeger. 'Gaussian processes for machine learning'. In: *International Journal of Neural Systems* 14.02 (2004), pp. 69–106 (cit. on p. 62).

[87] Daeun Song, Pierre Fernbach, Thomas Flayols, Andrea Del Prete, Nicolas Mansard, Steve Tonneau and Young J Kim. 'Solving footstep planning as a feasibility problem using l1-norm minimization'. In: *IEEE Robotics and Automation Letters (RA-L)* 6.3 (2021), pp. 5961–5968 (cit. on pp. 41, 42, 69, 85, 127, 136, 137).

[88] Manoj Srinivasan and Andy Ruina. 'Computer optimization of a minimal biped model discovers walking and running'. In: *Nature* 439.7072 (2006), pp. 72–75 (cit. on pp. 4, 105).

[89] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. 23rd May 2018. URL: https://www.ros.org (cit. on p. 88).

[90] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J. -. Laumond, L. Marchionni, H. Tome and F. Ferro. 'TALOS: A new humanoid research platform targeted for industrial applications'. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. 2017 (cit. on pp. 7, 68, 87, 132).

[91] T. Stouraitis, I. Chatzinikolaidis, M. Gienger and S. Vijayakumar. 'Online Hybrid Motion Planning for Dyadic Collaborative Manipulation via Bilevel Optimization'. In: *IEEE Transactions on Robotics (T-RO)* 36.5 (2020) (cit. on p. 29).

[92] Yuval Tassa, Tom Erez and Emanuel Todorov. 'Synthesis and stabilization of complex behaviors through online trajectory optimization'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 4906–4913 (cit. on pp. 16, 19).

[93] Steve Tonneau. 'Motion planning and synthesis for virtual characters in constrained environments'. PhD thesis. INSA de Rennes, 2015 (cit. on pp. 13, 15).

[94] Steve Tonneau, Andrea Del Prete, Julien Pettré, Chonhyon Park, Dinesh Manocha and Nicolas Mansard. 'An efficient acyclic contact planner for multiped robots'. In: *IEEE Transactions on Robotics (T-RO)* 34.3 (2018), pp. 586–601 (cit. on pp. 2, 12, 14, 15, 29, 105).

[95] Steve Tonneau, Pierre Fernbach, Andrea Del Prete, Julien Pettré and Nicolas Mansard. '2pac: Two-point attractors for center of mass trajectories in multi-contact scenarios'. In: *ACM Transactions on Graphics (TOG)* 37.5 (2018), pp. 1–14 (cit. on p. 37).

[96] Steve Tonneau, Daeun Song, Pierre Fernbach, Nicolas Mansard, Michel Taïx and Andrea Del Prete. 'SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019 (cit. on p. 14).

[97] Marc Toussaint, Kelsey Allen, Kevin Smith and Joshua Tenenbaum. 'Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning'. In: *Robotics: Science and Systems*. 2018 (cit. on pp. 2, 19, 29).

[98] Marc Toussaint, Kelsey R Allen, Kevin A Smith and Josh B Tenenbaum. *Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning – Extended Abstract*. Sister Conference Best Paper Track – Extended abstract of the R:SS'18 paper. 2019 (cit. on p. 19).

[99] Arun Venkatraman, Byron Boots, Martial Hebert and J Andrew Bagnell. 'Data as demonstrator with applications to system identification'. In: *ALR Workshop, NIPS*. 2014 (cit. on p. 19).

[100] Julian Viereck, Avadesh Meduri and Ludovic Righetti. 'ValueNetQP: Learned one-step optimal control for legged locomotion'. In: *Learning for Dynamics and Control Conference*. PMLR. 2022, pp. 931–942 (cit. on p. 18).

[101] Jiayi Wang, Iordanis Chatzinikolaidis, Carlos Mastalli, Wouter Wolfslag, Guiyang Xin, Steve Tonneau and Sethu Vijayakumar. 'Automatic gait pattern selection for legged robots'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 3990–3997 (cit. on p. 8).

[102] Jiayi Wang, Sanghyun Kim, Teguh Santoso Lembono, Wenqian Du, Jaehyun Shim, Saeid Samadi, Ke Wang, Vladimir Ivan, Sylvain Calinon, Sethu Vijayakumar and Steve Tonneau. 'Online Multi-Contact Receding Horizon Planning via Value Function Approximation'. In: *arXiv preprint arXiv:2306.04732* (2023) (cit. on p. 7).

[103] Jiayi Wang, Sanghyun Kim, Sethu Vijayakumar and Steve Tonneau. 'Multi-fidelity receding horizon planning for multi-contact locomotion'. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2021, pp. 53–60 (cit. on p. 7).

[104] Jiayi Wang, Teguh Santoso Lembono, Sanghyun Kim, Sylvain Calinon, Sethu Vijayakumar and Steve Tonneau. 'Learning to Guide Online Multi-Contact Receding Horizon Planning'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 12942–12949 (cit. on p. 7).

[105] Patrick M Wensing and David E Orin. 'Generation of dynamic humanoid behaviors through task-space control with conic optimization'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 3103–3109 (cit. on pp. 16, 18, 52).

[106] Patrick M Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard and Andrea Del Prete. 'Optimization-based control for dynamic legged robots'. In: *arXiv preprint arXiv:2211.11644* (2022) (cit. on p. 17).

[107] Pierre-Brice Wieber. 'Viability and predictive control for safe locomotion'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2008, pp. 1103–1108 (cit. on p. 16).

[108] Alexander Winkler, Ioannis Havoutis, Stephane Bazeille, Jesus Ortiz, Michele Focchi, Rüdiger Dillmann, Darwin Caldwell and Claudio Semini. 'Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 6476–6482 (cit. on pp. 12–14).

[109] Alexander W Winkler, C Dario Bellicoso, Marco Hutter and Jonas Buchli. 'Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization'. In: *IEEE Robotics and Automation Letters (RA-L)* 3.3 (2018), pp. 1560–1567 (cit. on pp. 2, 18, 19, 29, 52, 84).

[110] Alexander W Winkler, Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G Caldwell and Claudio Semini. 'Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 5148–5154 (cit. on pp. 12, 13, 15, 105).

[111] David A Winter. *Biomechanics and motor control of human movement*. John Wiley & Sons, 2009 (cit. on p. 106).

[112] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014 (cit. on p. 114).

[113] Weitao Xi, Yevgeniy Yesilevskiy and C David Remy. 'Selecting gaits for economical locomotion of legged robots'. In: *International Journal of Robotics Research (IJRR)* 35.9 (2016), pp. 1140–1154 (cit. on pp. 4, 106).

[114] Guiyang Xin, Wouter Wolfslag, Hsiu-Chin Lin, Carlo Tiseo and Michael Mistry. 'An optimization-based locomotion controller for quadruped robots leveraging Cartesian impedance control'. In: *Frontiers in Robotics and AI* 7 (2020), p. 48 (cit. on pp. 107, 127).

[115] Chuanyu Yang, Kai Yuan, Qiuguo Zhu, Wanming Yu and Zhibin Li. 'Multi-expert learning of adaptive legged locomotion'. In: *Science Robotics* 5.49 (2020) (cit. on p. 134).

[116] Seniha Esen Yuksel, Joseph N Wilson and Paul D Gader. 'Twenty years of mixture of experts'. In: *IEEE Transactions on Neural Networks and Learning Systems* 23.8 (2012), pp. 1177–1193 (cit. on p. 84).

[117] He Zhang, Sebastian Starke, Taku Komura and Jun Saito. 'Mode-adaptive neural networks for quadruped motion control'. In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), pp. 1–11 (cit. on p. 137).

[118] Mingyuan Zhong, Mikala Johnson, Yuval Tassa, Tom Erez and Emanuel Todorov. 'Value function approximation and model predictive control'. In: *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. 2013, pp. 100–107 (cit. on pp. 18, 31, 35).