OPTIMIZATION-BASED CONTROL AND PLANNING FOR HIGHLY DYNAMIC LEGGED LOCOMOTION IN COMPLEX ENVIRONMENTS

BY

YANRAN DING

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Abstract

Legged animals can dynamically traverse unstructured environments in an elegant and efficient manner, whether it be running down steep hill or leaping between branches. To harness part of the animal agility to legged robot would unlock potential applications such as disaster response and planetary exploration. The unique challenge of these tasks is that the robot has to produce highly dynamic maneuvers in complex environments with minimum human guidance. This thesis explores how optimization-based method can be applied in the control and planning of highly dynamic legged motions to address the locomotion problem in complex environments. Specifically, this work first describes the design synthesis of a small and agile quadrupedal robot *Panther*. Based on the quadruped platform, we developed a model predictive control (MPC) control framework to realize complex 3D acrobatic motions without resorting to switching among controllers. We present the MPC formulation that directly uses the rotation matrix, which avoids the singularity issue associated with Euler angles. Motion planning algorithms are developed for planar legged robot traversing challenging terrains. Dynamic trajectories that simultaneously reason about contact, centroidal dynamics, and joint torque limit are obtained by solving mixed-integer convex programs (MICP) without requiring any initial guess from the operator. We further reduce the computational expense of long-horizon planning by leveraging the benefits of both optimization and sampling-based approaches for a simple legged robot. Finally, we present experiment results for each topic on legged robot hardwares to validate the proposed method. It is our hope that the results presented in this thesis will eventually enable legged robots to achieve mobility autonomy at the level of biological systems.

To my family.

Acknowledgments

I would like to express gratitude to my supervisor Dr. Hae-Won Park for giving me the opportunity to pursue a Ph.D. in robotics. Thank you for the personal investment in my academic journey and constantly pushing me towards a higher level. I was able to pursue my ideas while receiving continuous guidance and inspirations. I would also like to thank Dr. Jo*ã*o Ramos, whose encouragement helped me through many ups and downs of research. I find myself very fortunate to have him as a mentor and as a friend during my time at UIUC. I am also grateful to two professors that I respect very much. Dr. Kris Hauser supported my research by providing his valuable advice through weekly research meetings. Dr. Patrick Wensing often makes insightful remarks that shed light on my research. It is my great fortune and honor to receive mentorship from these outstanding professors.

The work in this thesis would not have been possible without the collaborative work from all the great colleagues that I have the pleasure to work with. I want to thank Abhishek Pandala, Chuanzheng Li and Mengchao Zhang for their close collaborations that crystallized many inspiring ideas into fruitful results. Many thanks to Jaejun Park, Won Dong Shin, Jason Jeong, Yinai Fan, Jifei Xu, Young-Ha Shin and members of the HUBO lab during summer 2019. Thank you for making my graduate school life meaningful and memorable. I extend my gratitude to Dr. Zherong Pan for his helpful advice, and special thanks Dr. Jonathan Hoff for his being an incredible role model to academic survivors including myself. Thank you also to the MechSE staff and Professors, specially Gary Sedberry for his many advice on machining, Prof. Geir Dullerud for his course on convex method in control.

Finally, I want to thank my friends that I met along the way, Yiliang, Yichuan, Mihary, Ben, Nick, Ophelia, Mengwei, Gang, aunt Shuqin, uncle Lianghu and others. You have made this journey unforgettable and special. Most importantly, I want to thank my family for their bountiful love and support. My caring parents Binliang Ding and Jian Ye, my nourishing grandparents, my loving wife Xian who always has unshakable belief in me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Legged animals possess extraordinary competence in negotiating complex unstructured environments by executing well-coordinated movements. Whether it be mountain goats scaling steep cliffs, brown bears climbing trees or dogs trained to perform Parkour motions, quadrupedal animals display remarkable capabilities well beyond those of current legged machines. In particular, nimble animals such as squirrels are extremely adept at dynamic maneuvers such as climbing up a tree and leaping between the branches, as shown in Fig. 1.1. To transfer part of the agility of these animals to legged robots would open up numerous applications including disaster response, transportation and space exploration.

The remarkable mobility of agile animals inspired the development of novel actuation schemes that gave birth to a new generation of legged robot hardware [11, 47, 60, 67, 70, 107, 120, 125]. These platforms possess actuation capabilities comparable to or even exceeding their animal counterparts. In addition, the advancements in numerical algorithms and machine learning have empowered the implementation of increasingly dynamic legged locomotions [29, 44, 72, 108, 147, 148, 150, 154]. One of the goals shared by many researchers is to deploy the robots in environments that are too dangerous for humans, such as disaster response scenarios and planetary exploration missions. Situations such as destroyed buildings and rocky cliffs present challenging problems for robots that rely on quasi-static locomotion strategy or a limited set of pre-computed motions. In particular, these situations require the robot to carry on a task while the human operator has limited information about the

Figure 1.1: A squirrel jumping from a tree branch [96]

environment with considerable transmission delay. Hence, the robot has to produce highly dynamic maneuvers for complex path planning problems (semi-)autonomously. Currently, robot systems that can rival the capabilities of biological systems in these unstructured environments have not yet been developed. The major challenges that hinder the endeavor to reproduce these capabilities in robotic systems include (1) model complexities that are fundamental to legged locomotion, such as high degrees of freedom (DoF), under-actuation, and hybrid/nonlinear dynamics; (2) challenges in the design of a motion planning framework that simultaneously exploits the complex dynamic properties of legged robot systems and the combinatorial richness in the contact with environment; and (3) lack of understanding of the physical system design essential for the execution of dynamic motions. In particular, state-of-the-art motion planning algorithms for legged robots are not well suited for such an application because (a) many works focus on stabilizing dynamic periodic locomotion, which is not applicable to complex environments; (b) algorithms tailored for complex environments often require a good initial guess of the robot trajectory from the heuristics of a human operator; (c) many algorithms involve long computational time due to the expensive operations in a high-dimensional search space.

The objective of this thesis is to make contributions to address some of the critical problems in these challenges. Particularly, connection will be established among the control of legged robot systems, motion planning strategies, and the hardware system synthesis principles to create a systematic solution that enables legged robots to negotiate complex environments.

1.2 Related Work

The area of dynamic legged locomotion has progressed rapidly thanks to the enormous amount of research on motion control and planning for an arsenal of robots. A rough categorization of the related work in these areas is presented in this section. Reactive control regulates system dynamics at the current moment while the model predictive control approach plans motion within a short prediction horizon. The optimization- and sample-based motion planning approaches have longer behavior horizon foresight and have been often applied with reactive controllers for successful locomotion.

1.2.1 Reactive Control for Locomotion

The reactive control regulates the instantaneous system dynamics around a reference trajectory which is computed a priori. Two of the widely adopted reactive control approaches in legged locomotion are inverse dynamics control and operational-space control. Inverse dynamics control can calculate the joint torque of a high degree-of-freedom (DoF) robot at runtime [41, 58, 92, 115]. The disadvantage of this approach is that it has no authority to alter the pre-designed trajectory, which can result in failure due to torque limit or violation of the friction-cone constraint.

Operational-space control [71,121] or task-space control focuses on regulating the motion in the operational-space instead of the joint space as in inverse dynamics control. The switching contact and under-actuation inherent to legged systems can be resolved [61,93,116,141] and system redundancy can be exploited to achieve multiple objectives in a hierarchical manner [61,122,123,140]. The myopic nature of the reactive controller requires high-level controllers with longer behavior horizon foresight for successful locomotion. Chapter 2 presents an application of a Quadratic Program (QP)-based reactive controller and trajectory optimization for dynamic squat jumping on a quadrupedal robot.

1.2.2 Model Predictive Control (MPC)

Enabling agile locomotion capability in legged robot systems requires the control to utilize inherent dynamics while handling constraints from hardware limitations and interactions with the environment. Model Predictive Control (MPC) recently became a widespread control method due to recent advancements in computing hardware and optimization algorithms, which enabled real-time execution of the MPC controller in embedded systems. The major difference between reactive control and MPC is that the former applies control in a myopic manner while the latter plans control within the span of a prediction horizon, which exploits system dynamics in a more graceful manner. Based on a model prediction, the MPC framework easily incorporates various constraints by transcribing the control law as an optimization problem. Recent applications of MPC on humanoids [54], [51] and quadrupeds [100] have shown the capability of MPC in planning and controlling complex dynamic motions while embracing system dynamics and constraints arising from friction and motor saturation.

Despite the widespread adaptation of MPC, its direct implementation on a high degree-of-freedom (DoF) system requires heavy computational resources, hindering the application on embedded platforms. To tackle this problem, simpler models or templates [43] that capture the dominant system dynamics were used to predict the behavior of the system. Previous MPC schemes [54], [55] worked on simplified dynamics models such as Linear Inverted Pendulum [65] (LIPM) to facilitate online execution. The planar single rigid body model is used in [109] to plan online jumping trajectories for MIT Cheetah 2 with different obstacle heights. The spring-mass model is used in [150] to achieve jumping and landing. Thes centroidal dynamics [103] model links the linear and angular momentum of the robot with the external wrench. This model is used in [75], [26] to capture the major dynamic effect of the complex full-body dynamics model of the humanoid robot Atlas. Recent work [10,12, 29] use centroidal dynamics in the MPC to achieve various dynamic gaits in quadrupedal robots.

1.2.3 Motion Planning for Locomotion

The two dominant methods in motion planning for locomotion are the optimization-based approach and the sample-based approach. A brief overview of these two approaches is provided here.

Optimization-based Motion Planning

The optimization-based motion planning, or trajectory optimization, uses optimization to generate dynamically feasible trajectories, as originally introduced by Witkin and Kass [144]. The decision variables are finite number of robot motion parameters; the optimization constraints enforces collision avoidance and physical law (e.g. system dynamics, kinematics, actuation limit). The objective function rewards desirable behaviors and sometimes penalize undesirable ones. There are many methods to formulate a trajectory optimization problem [6, 7, 57, 95, 98, 112, 137] including single shooting, multiple shooting, direct collocation, differential dynamic programming and through-contact methods. Most of these transcriptions result in a nonlinear program, which can be solved by numerical solvers [3, 9, 18, 138] to synthesize smooth dynamical motions. Trajectory optimization has been widely applied to humanoids [23, 26, 56, 59, 131] and quadrupeds [67, 86, 99, 108, 143]. For simple robot models, the optimization problems can be convex and solved efficiently [15]. However, complicated robot model can introduce nonlinearity and non-convexity and cause the optimization to give local minima or infeasibility [4]. To tackle this problem, Chapter 4 proposes a mixed-integer program-based method to solve the non-convex problem and generate certificates for global optimality or infeasibility.

Sample-based Motion Planning

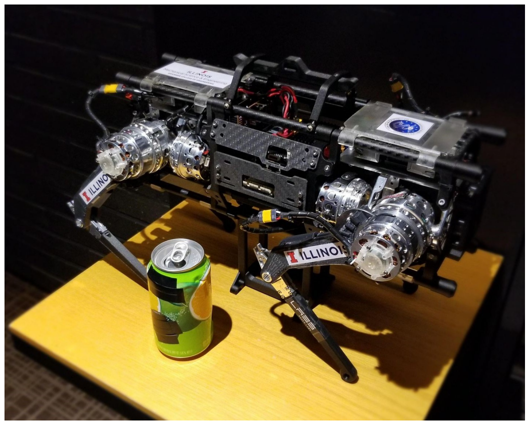
The sample-based motion planning method [78] draws a random sample and then determines if the sample is feasible. A path is found when the connected feasible samples include start and goal. Popular sample-based motion planning techniques such as Probabilistic Roadmap (PRM) [14, 68] and Rapidly-exploring Random Trees (RRT) [77, 79] can guarantee probabilistic completeness. The sample-based method has been widely used to solve large planning problems in humanoid [49, 50, 73, 74] and multi-legged robots [16, 118, 126]. However, most of these methods are based on quasi-static assumption, and require custom post-processing to improve the quality of the motion. Inspired by [126], Chapter 5 introduces a hybrid sample/optimization-based planning framework for agile jumping robots.

1.3 Organization and Contributions

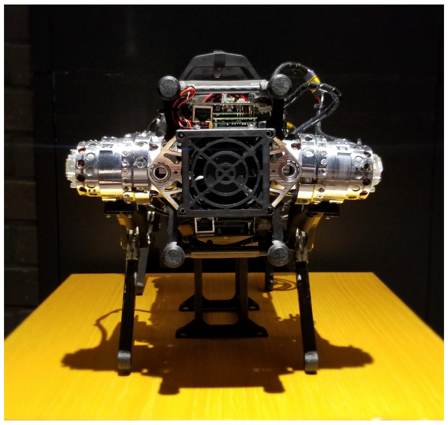
This thesis contributes to the advancement of dynamic legged locomotion, with more specific contributions including high-power actuator design, model predictive control, and kinodynamic motion planning for legged robots to traverse complex terrains by executing dynamic maneuvers. The chapters in this thesis are organized in a *bottom-up* order, where hardware design is first introduced, then the real-time controller before the high-level motion planning algorithms.

Chapter 2 describes the hardware synthesis of the legged robot platforms used in Chapters 3-5. A small yet agile quadrupedal robot called *Panther* as shown in Fig. 1.2 is developed because platforms capable of the dynamic motions we want to achieve were not available at that time. Details about actuator design and the electronic system are also presented since they enable the various hardware experiments in this thesis. To showcase the legged robot hardware capabilities, a dynamic squat jumping experiment on *Panther* is presented.

Chapter 3 presents a novel Representation-Free Model Predictive Control (RF-MPC) framework for controlling various dynamic motions of quadrupedal robots in three-dimensional



(a)



(b)

Figure 1.2: Pictures of *Panther*, a small and agile quadrupedal robot platform used in this thesis. (a) The isometric view of *Panther* with a soda can for scale (b) The front view of *Panther*.

(3D) space. Our formulation directly represents the rotational dynamics using the rotation matrix, which eliminates the issues associated with the use of Euler angles and quaternion as the orientation representations. With a variation-based linearization scheme and a carefully constructed cost function, the MPC control law is transcribed to the standard Quadratic Program (QP) form. Operating at a real-time rates of 250 Hz on the quadruped robot *Panther*, the MPC controller enables dynamic motions with arbitrary orientation using a single control framework without resorting to switching among controllers. Having a unified framework is beneficial because switching between controllers is either slow or prone to edge case failure. Experimental results including periodic gaits and a controlled tumble are presented to validate that RF-MPC can stabilize dynamic motions that involve singularity in 3D maneuvers.

Chapter 4 introduces a novel mixed-integer convex programming (MICP) formulation for kinodynamic motion planning of dynamic legged robots. The proposed MICP-based planner can produce motions that exploit the environment by simultaneously reasoning about the centroidal dynamics, actuator torque limit, contact position, and gait sequence. Specifically, the non-convex torque limit constraint is reformulated as a piece-wise convex constraint over a discretization of the configuration space (C-space) of the robot. Compared with nonlinear program-based methods, the MICP formulation provides a globally optimal solution using off-the-shelf numerical solvers without requiring initial guesses. Simulation and experiment results on multiple-legged robots are presented to validate the proposed method.

Chapter 5 presents a hybrid sampling/optimization - based motion planning algorithm for dynamic single-legged robots to overcome challenging terrains. The kinodynamic motion planning problem is decoupled into two stages, where the sampling stage searches for a kinematically feasible path as a sequence of parabolas, and the optimization stage solves for the dynamically feasible trajectory. The performance of the proposed hybrid motion planning algorithm is shown on various example terrains, and the advantage of this method is highlighted through benchmarking with two other methods. A trajectory generated by the proposed method is applied on a physical robot, which successfully traversed a challenging terrain by executing 3 consecutive jumps.

Chapter 6 provides the concluding remarks and introduces the possible future research directions from this thesis.

Chapter 2

Robotic System Synthesis

The design guidelines for dynamic legged robots can be acquired by observing nimble animals such as squirrels. For example, leg inertia should be reduced by using light-weight components, and heavy parts such as actuators should be placed close to the body. This design strategy allows for fast swing leg motion and reduces dynamic coupling between swing legs and the body, therefore admits a simple model for control and planning. In addition, the actuators should be power-dense and compliant so that they can propel the robot to perform dynamic maneuvers while frequently interacting with the environment.

This chapter outlines how the design guidelines are embodied in the synthesis process of a quadrupedal robot *Panther* as shown in Fig. 1.2. The overall description and mechanical properties of *Panther* are presented as well as the actuator design process. Details about the electronic framework, essential robot software, and practical implementation are also provided for completeness. The capability of *Panther* is demonstrated in a power squat jump experiment where the robot reached a maximum jumping height of 0.7 m (2.25 $\times$ body length) and landed safely.

2.1 Panther - a Small and Agile Quadruped Robot

This section presents *Panther*, a 5.5 kg fully torque controllable, electrical quadruped robot. With a body length of 0.3 m and link length of 0.14 m, this robot has the overall dimension of a slightly oversized domestic cat. Compared with other state-of-the-art quadruped robot platforms in the same category, *Panther* is the smallest in size and the lightest in weight.

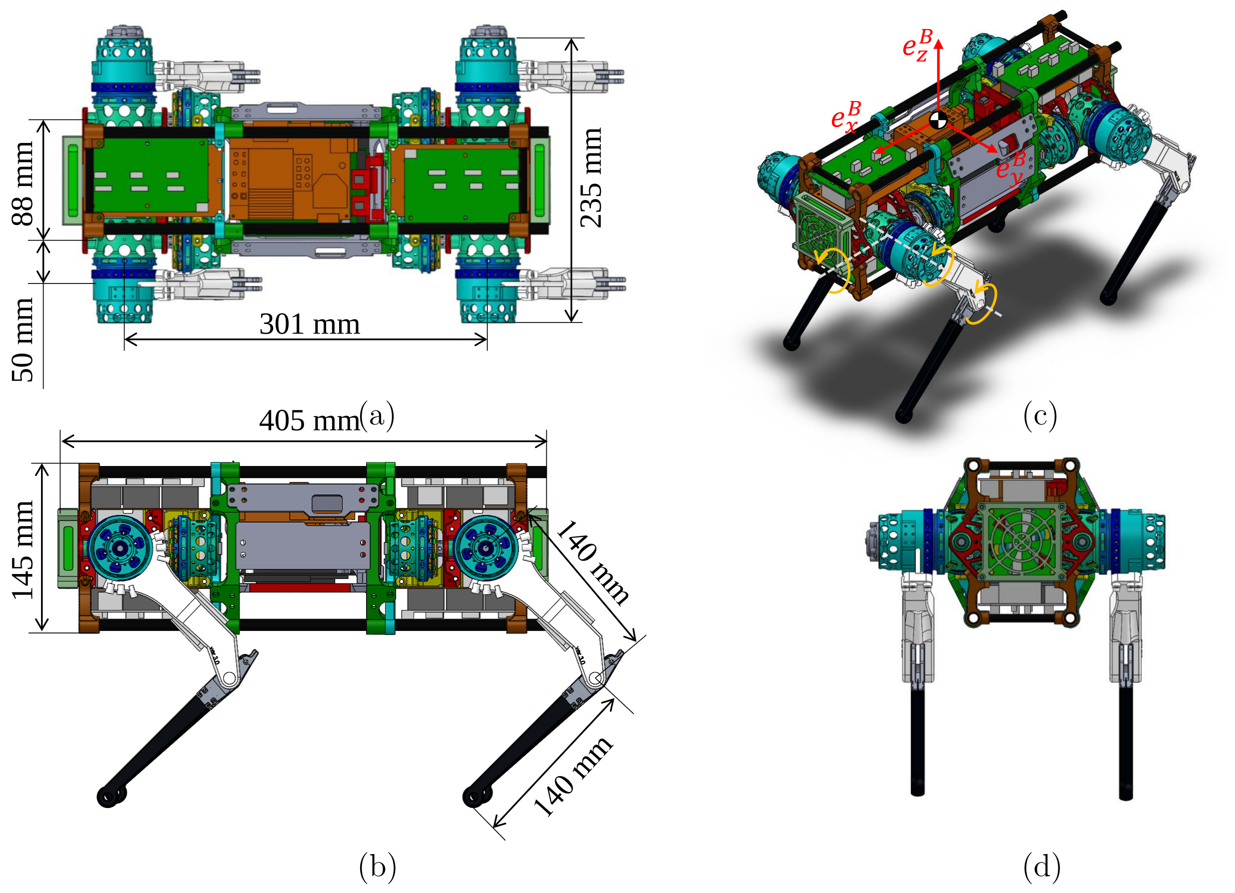


Figure 2.1: The quadruped robot platform *Panther* has four proprioceptive leg modules with high specific power capability. Each leg module consists of one ABAD, HIP and KNEE motor modules. The dimensions and weight of Panther resembles that of a large domestic cat. (a) top view (b) side view (c) isometric view (d) back view.

Table 2.1: System Parameters of *Panther*

Parameter	Value	Unit
total mass	5.500	kg
body length (motor2motor)	0.301	m
linkage length (thigh&shin)	0.140	m
enclosing dim. (L×W×H)	0.405×0.235×0.145	m
I_{xx}	0.026	kg·m^2
I_{yy}	0.112	kg·m^2
I_{zz}	0.075	kg·m^2
max. ABAD torque	8.410	N·m
max. HIP/KNEE torque	9.811	N·m
max. ABAD speed	79.800	rad/s
max. HIP/KNEE speed	33.400	rad/s

The small-dimension and light-weight design grants the robot inherent robustness and agility. The downside of a small-scaled robot is its lower payload capacity. Specifically, the operating time that the on-board battery can sustain is lower than larger robots (around 20 minute). Nevertheless, since the main focus of this thesis is on dynamic motions, agility and robustness are the most important performance metrics.

2.1.1 Dimension

The size and dimensions of *Panther* are chosen to resemble that of a domestic cat. Fig.2.1 presents a CAD drawing of the robot with important dimensions. Four leg modules with proprioceptive actuation design [142] are arranged in a left-right symmetric manner, knees bending backward. The proprioceptive motor design paradigm provides high torque density and high-bandwidth force control that is suitable for dynamic locomotion. With carbon fiber tubes providing structural rigidity, the electronic components are enclosed in the middle of the body. To achieve dynamic locomotion capability, *Panther* is designed to have a high strength-to-weight ratio. For example, carbon fiber reinforced 3D printed parts are extensively used in body assembly. To provide the basic specifications of the *Panther* robot, important system parameters are presented in Table 2.1.

The main enabler for *Panther* to perform highly dynamic motions is the proprioceptive actuator design and light-weight leg design. The high specific-torque actuator is designed based on the proprioceptive actuation paradigm [124], whose detailed design procedure is presented in Section 2.2. The light-weight linkage design is another factor that ensures that the robot can interact with the environment in a compliant manner. The leg module synthesis is detailed in Section 2.1.3.

2.1.2 Main Body Design

The main body of *Panther* is designed to provide structural integrity and storage for the electronic components. As shown in Fig. 2.1, four carbon tubes run along the longitudinal direction of the robot to provide structural stiffness. Many parts used in the main body assembly are 3D printed Nylon with carbon fiber reinforcement (Onyx) for its high strength-to-weight ratio. These parts are manufactured using the Mark Two$^{\text{TM}}$ 3D printer. In addition, water-jetted carbon fiber panels are installed on the two sides to shield the electronics and provide extra stiffness. The body design allows the robot to be placed vertically, which makes it easy to access the electronic components for repair purposes. The body frame $\{B\}$ is located at the center of mass (CoM) of the robot, which is shown in Fig. 2.1(c).

The main body is also carefully designed to accommodate the electronics. The main computer, power distribution board, and an inertia measuring unit (IMU) are mounted on a 3D printed part with plastic spacers in between for shock isolation, and the whole assembly is installed in the cavity of the body. The Elmo$^{\text{TM}}$ amplifiers are arranged into groups of three, and they are mounted symmetrically in an up-down / front-back fashion, where each Elmo$^{\text{TM}}$ amplifier group is in charge of controlling one leg module. This arrangement makes the ports on the Elmo$^{\text{TM}}$ boards more accessible for the calibration process. The Elmo$^{\text{TM}}$ heat sink is attached via thermal tape to the other aluminum parts, which serve as heat sinks for heat dissipation. Furthermore, a fan is attached to each end of the robot for active heat

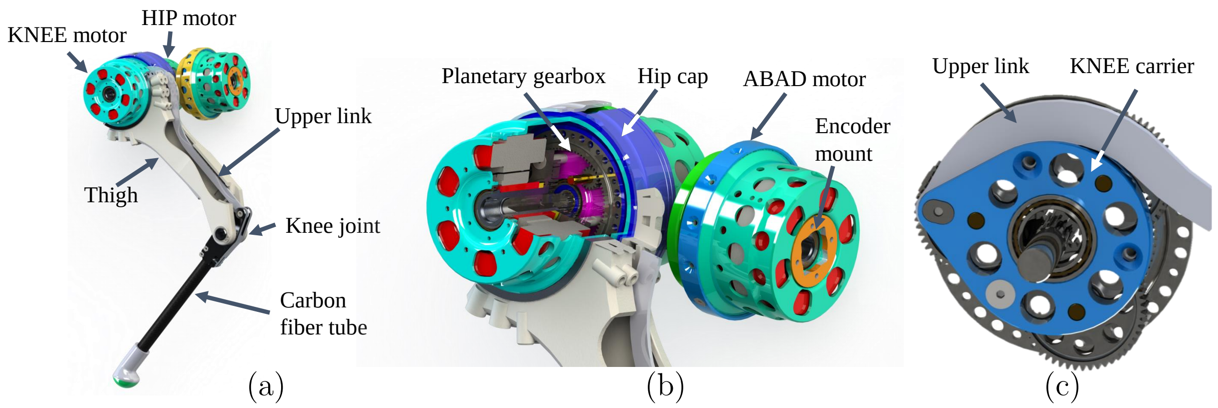


Figure 2.2: CAD rendering of the leg module and the planetary gearbox (a) The motor configuration and the linkage design (b) Cross-section view of the HIP-KNEE module, showing the planetary gearbox and motor (c) A zoom-in view on the curved upper link and the KNEE carrier.

exchange, which is necessary because Elmo generates a considerable amount of heat when the robot is executing dynamic motions. More details about the electronic components can be found in Section 2.3.

2.1.3 Leg Module Synthesis

Each leg module is composed of three brushless direct-current (BLDC) motor modules, namely, HIP, KNEE, and ABAD (abduction/adduction) motors. An illustration of the leg module is presented in Fig. 2.2. The ABAD motor axis is aligned with the x-axis of the body frame $\{B\}$, and the HIP and KNEE motor modules are placed coaxially. It is worth noting that the hip cap in Fig. 2.2 (b) connects the KNEE motor to the HIP motor via a large diameter bearing. This design allows the hip joint to rotate 360°, permitting *Panther* to walk even when the body is up-side-down.

The torque from the KNEE motor is transmitted to the knee joint via a curved upper link, which is made of hardened A2 tool steel for extra strength. As shown in Fig. 2.2 (c), the top part of the upper link is curved so that the knee carrier can rotate an extra 30° compared with straight link design. The hollow design of the 3D printed thigh link allows the upper link to go through, providing protection from external impact. The thin-wall thigh link and the carbon fiber tube shank link constitute a light-weight articulated leg, whose weight is less than 10% of the total weight of the leg module. This design feature allows the subsequent motion planning and control design to adopt the mass-less leg assumption. This is a key assumption since it facilitates the employment of a simple model, which greatly alleviates the computational burden in the corresponding optimizations.

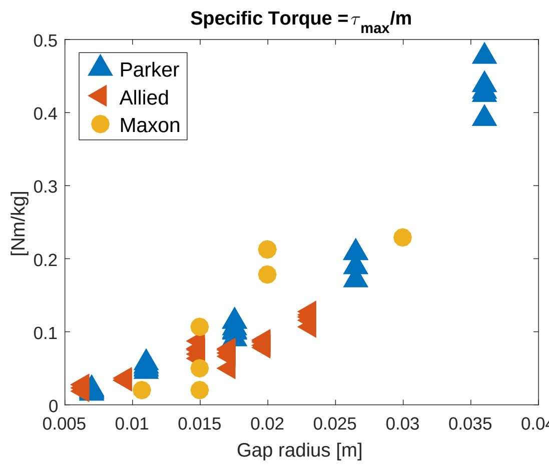
The point foot of the leg module shown in Fig. 2.2 (a) is made of a Nylon 3D printed part cushioned with Sorbothane for its shock absorption capability.

2.2 Actuator Design

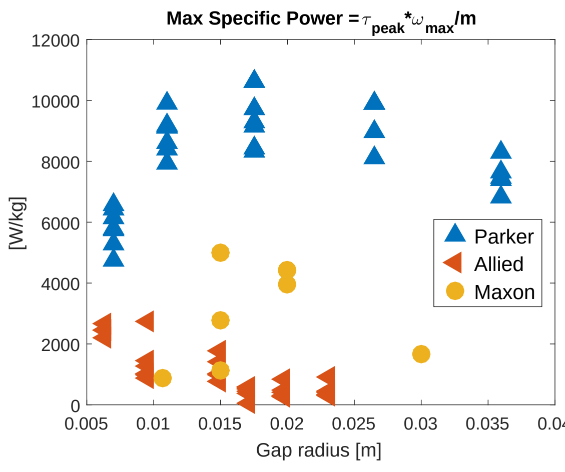
Dynamic maneuvers such as jumping require the actuator to output large torque at a high angular speed. To design an actuator that is suitable for dynamic locomotion application, torque density [125] was proposed to be the deciding metric because it reflects the ratio of torque-producing capability versus the weight of an actuator. This design strategy is successful in MIT Cheetah I, II, III [124], [107], [11]. However, as the scale of the quadrupedal robot decreases, motor speed becomes more of a limiting factor since higher angular velocity is needed to produce the same end-effector speed. Based on this observation, the major design consideration is set to be the balance between motor torque and speed for small dynamic quadruped robots.

2.2.1 Motor Selection

The motor selection strategy adopted here is to strike a balance between torque and speed requirements. The maximal specific power $\tau_{max} \cdot \omega_{max}/m$ takes into account both torque and speed-producing capabilities of the motor. It also indicates the power delivery ability per unit weight in jumping applications. Moreover, since the maximal specific power is independent of the gear ratio, it can be used to select the motor before determining the gear ratio. Although simultaneously considering the motor selection and gear ratio can be



(a)



(b)

Figure 2.3: Specifications of BLDC motors from Parker, Allied Motion, and Maxon Group [88]. (a) Maximum specific torque versus gap radius. (b) Maximum specific power versus gap radius.

an interesting problem, here we adopt the sequential design approach, where the motor is chosen before the gear ratio is determined for the lower computational cost. Nevertheless, the motor module is validated experimentally to be competent as shown in Section 2.6. In conclusion, the maximal specific power is chosen as the metric for motor selection.

The maximal specific torque data of commercial motor are plotted in Fig.2.3(a) against gap radius. The gap radius is the "distance from the rotating axis to the center of the gap between permanent magnets and the stator" [124]. The maximum specific power data are plotted against gap radius in Fig.2.3(b). While specific torque is close among motors with similar gap radius, Parker^{TM} motors have the highest maximal specific power value under the same gap radius. Taking into account the dimensional limit of motor installation on a small-sized targeting robot platform (OD<50 mm; stack length<50 mm), Parker^{TM} motor K044050 was chosen for HIP and KNEE joints and K044025 was chosen for the ABAD joint.

2.2.2 Determining Gear Ratio

To determine the gear ratio of the gearbox, we presented in [34] a sequential design procedure with the assistance of nonlinear program (NLP). Here we briefly go over the major steps and formulations for completeness. Further details can be found in the companion design paper [34].

At stage one, an NLP is formulated where the robot is simplified as a point-mass moving along a vertical linear rail. Joint torque and other constraints are imposed and the maximal jumping height is maximized. The gear ratio of stage one (GR_1) is used as an initial guess in stage two, where a more realistic model is used.

At stage two, approximate joint inertia (as a function of gear ratio) is incorporated in the model so that impact force is present. The optimization variable $\boldsymbol{x}_{opt}$ is

$$\boldsymbol{x}_{opt} := [\boldsymbol{p}_0, \boldsymbol{\alpha}_F, T_{st}, GR, \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}, \boldsymbol{q}^-], \tag{2.1}$$

where $\boldsymbol{p}_0$ is the intial CoM position; $\boldsymbol{\alpha}_F$ is the coefficient for the Bézier polynomials that parametrize the GRF; GR is the gear ratio; $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}$ are the joint position, velocity and torque trajectories, respectively; $\boldsymbol{q}^-$ is the joint angle at touchdown. The variables $\boldsymbol{p}_0, \boldsymbol{\alpha}_F, T_{st}$ characterize the jumping trajectories, GR modulates the actuator box constraints, and $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}, \boldsymbol{q}^-$ are slack variables for imposing path constraints.

The NLP formulation at stage two is

$$\begin{aligned}
\underset{\boldsymbol{x}_{opt}}{\text{minimize}} \quad & -\frac{h_{max}}{||\boldsymbol{F}||} && \text{(2.2a)}\\
\text{subject to} \quad & C_{impact}(\boldsymbol{x}_{opt}) = 0 && \text{(2.2b)}\\
& C_{kinematics}(\boldsymbol{x}_{opt}) = 0 && \text{(2.2c)}\\
& C_{box}(\boldsymbol{x}_{opt}) \leq 0, && \text{(2.2d)}
\end{aligned}$$

where the objective of the second NLP is to maximize the ratio of maximum jumping height and impact force norm $\frac{h_{max}}{||\boldsymbol{F}||}$ because it is desirable to have a large maximal reachable height and smaller impact force. The impact constraint C_{impact} is imposed to reveal the impact force

$$C_{impact} = \boldsymbol{D}(\dot{\boldsymbol{q}}^{+} - \dot{\boldsymbol{q}}^{-}) - \boldsymbol{J}^{T}(\boldsymbol{q}^{-})\boldsymbol{F} + \boldsymbol{J}_{x}^{T}(\boldsymbol{q}^{-})F_{x}, \tag{2.3}$$

where $\boldsymbol{D}(GR) \in \mathbb{R}^{4\times4}$ is the inertia tensor; $\boldsymbol{q}^{-} \in \mathbb{R}^{4}$ is the joint angle at impact; $\dot{\boldsymbol{q}}^{-}, \dot{\boldsymbol{q}}^{+} \in \mathbb{R}^{4}$ are the joint velocity before and after impact, respectively. $\boldsymbol{F} \in \mathbb{R}^{2}$ is the impact force vector from ground to the foot; $F_x \in \mathbb{R}$ is the horizontal impact force from the linear rail to the leg base; $\boldsymbol{J} \in \mathbb{R}^{2\times4}$ is the foot Jacobian; $\boldsymbol{J}_x \in \mathbb{R}^{1\times4}$ is the Jacobian of the base horizontal position. The constraint $\boldsymbol{J}\cdot\dot{\boldsymbol{q}}^{+} = \boldsymbol{0}$ ensures that the foot maintains contact with the ground; the constraint $\boldsymbol{J}_x\cdot\dot{\boldsymbol{q}}^{+} = 0$ makes sure the base does not break away from the linear rail. By solving the equations simultaneously, $\dot{\boldsymbol{q}}^{+}$,$\boldsymbol{F}$, and F_x can be expressed as

$$\begin{bmatrix} \dot{\boldsymbol{q}}^{+} \\ \boldsymbol{F} \\ F_x \end{bmatrix} = \begin{bmatrix} \boldsymbol{D} & -\boldsymbol{J}^{T} & -\boldsymbol{J}_{x}^{T} \\ \boldsymbol{J} & 0 & 0 \\ \boldsymbol{J}_x & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{D}\dot{\boldsymbol{q}}^{-} \\ 0 \\ 0 \end{bmatrix}, \tag{2.4}$$

$C_{kinematics}$ is the constraint that ensures consistency in forward kinematics and differential kinematics, and $C_{box} \leq 0$ represents the box constraints on $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}$

$$\begin{aligned} q_{min} \leq& \boldsymbol{q}_k \leq q_{max} \\ \dot{q}_{min}(GR) \leq& \dot{\boldsymbol{q}}_k \leq \dot{q}_{max}(GR) \\ \tau_{min}(GR) \leq& \boldsymbol{\tau}_k \leq \tau_{max}(GR), \forall k \end{aligned} \tag{2.5}$$

where the subscripts $(\cdot)_{min}$ and $(\cdot)_{max}$ stand for the lower and upper bounds, respectively. The trajectory optimization (TO) transcription is a polynomial-parametrized single-shooting with path constraints at sampled points. Compared with other transcription methods such as direct collocation, this method was adopted for its simplicity to implement. Since this problem is relatively easy to solve, the presented TO formulation can find solutions by using the GR_1 from the stage one to warm-start the search. Specifically, the value of the stage two gear ratio is obtained GR_2=23.36:1 by solving the NLP using $fmincon$ in MATLAB.

In the first two stages, the gear ratio is assumed to be a continuous variable, which is not realistic since the discrete gear teeth number choices result in a discrete gear ratio. Section 2.2.3 presents the more practical aspect of the gearbox design.

2.2.3 Planetary Gearbox Design

Speed reduction systems widely used in the robotics community include harmonic drive, cycloidal gearbox, and planetary gearbox. The planetary gearbox design is adopted here because its nominal gear ratio is lower than the other two designs and its efficiency is higher. Lower gear ratio is desirable here because it provides better transmission transparency or *back-drivability*, which is essential in the proprioceptive actuator design. A planetary gearbox consists of sun gear, a ring gear, a carrier, and typically multiple planet gears. This section presents the design of a two-stage *compound* planetary gearbox, which provides the desired gear ratio while maintaining a compact design.

Gear Ratio

In this section, the gear ratio (GR) of a compound planetary gearbox is derived as a function of the gear teeth number. To facilitate the derivation, the schematics of a compound planetary gearbox are presented in Fig. 2.4. The pitch radius of each component is denoted as r, the gear teeth number as N, and the angular velocity as ω. The subscripts and their referred components are: s (sun), p (planet), r (ring), c (carrier). The gear ratio is

$$GR = \frac{\omega_s}{\omega_c} = \frac{r_c}{r_s} \cdot \frac{r_{p1} + r_{p2}}{r_{p2}} = \frac{N_c}{N_s} \cdot \frac{N_{p1} + N_{p2}}{N_{p2}}, \tag{2.6}$$

where the subscripts p_1, p_2 refer to the two stages of the planet gear. As shown in Fig. 2.4, p_1 is green and p_2 blue. The ratio of pitch radius is the same as that of teeth number because the module number of all of the gears are the same.

The expression of gear ratio is deduced from the velocity matching conditions at the two

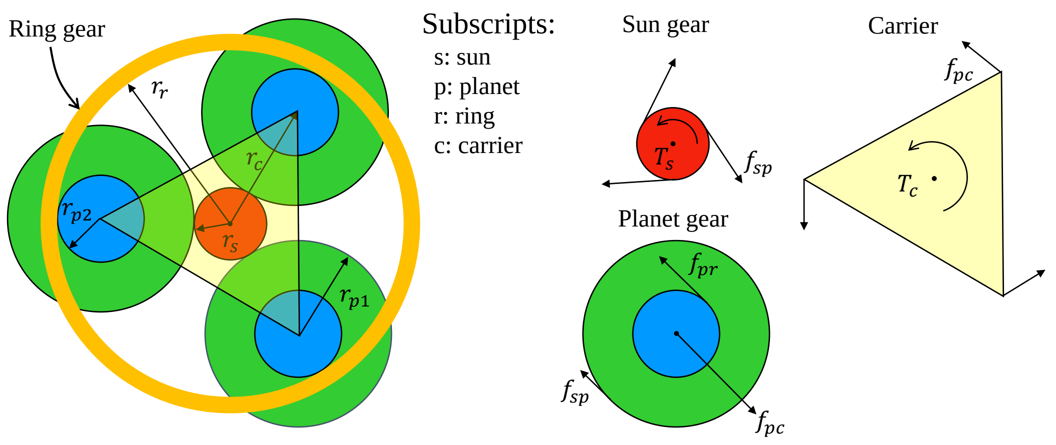


Figure 2.4: Schematics of the compound planetary gearbox with three planet gears. The sun gear is colored red; the planet gear is green (p1) and blue (p2); the carrier is represented by the transparent triangle. The ring gear is colored orange and assumed to be fixed. The free-body-diagrams of the sun gear, planet gear, and carrier is shown on the right for the derivation of the rotary inertia.

gear meshing points. One between the sun gear and the planet gear p_1, the other between the planet gear p_2 and the ring gear.

$$\begin{aligned} \omega_s \cdot r_s &= \omega_p(r_{p1} + r_{p2}) \\ \omega_c \cdot r_c &= \omega_p r_{p2}, \end{aligned} \tag{2.7}$$

where $r_c = r_s + r_{p1}$.

Gear Teeth Number

In Section 2.2.2, the gear ratio is assumed to be a continuous variable in the NLP. This is not realistic because gear teeth number can only take discrete values, hence, the desired gear ratio usually can not be achieved exactly. Therefore, given the desired gear ratio, the gear teeth number should be chosen such that the resulting gear ratio is as close to the desired gear ratio as possible. In addition, other design requirements such as dimension constraints should be taken into consideration. The problem of choosing the optimal set of gear teeth can be solved by the brutal-force method. This method is time-consuming because it enumerates all of the possible combinations of gear teeth numbers.

To find the optimal gear teeth combination faster, an integer program formulation is proposed. The optimization variables are $N_s, N_{p1}, N_{p2} \in \mathbb{Z}_+$, which refer to the gear teeth number of the sun gear and planet gear (stage 1 and 2), respectively. The gear ratio GR is a function of the optimization variables, the expression of the function is shown in (2.6). The optimization problem is formulated as an integer program as follows

$$\begin{aligned} \underset{N_s, N_{p1}, N_{p2}}{\text{minimize}} \quad & |GR - GR_d| \\ \text{subject to} \quad & N_s, N_{p1}, N_{p2} \in \mathbb{Z}^+ \\ & r_i \in [\underline{r}_i, \bar{r}_i], i \in \{s, p1, p2\} \\ & r_s + 2r_{p1} \leq r_{max} \end{aligned} \tag{2.8}$$

where GR_d is the desired gear ratio; $\underline{r}_i$ and $\bar{r}_i$ denote the lower and upper bound on the gear radius, respectively; the distance from the center of the gearbox to the farthest point of the planet gear is $r_s + 2r_{p1}$; r_{max} is the maximum radius constraint imposed by the dimension of the gearbox cover. This integer program is formulated using YALMIP [83], and solved using the branch-and-bound (B&B) algorithm. Compared with the brutal-force enumeration, the integer program approach is much more computationally efficient and easier to implement. For a small problem, like the one presented here, the optimal gear teeth combination can be solved in a few seconds, whereas the brutal-force method may take up to several minutes. In addition, the integer program approach makes it easy to incorporate additional parameters and constraints. For example, the gear module number can be incorporated as part of the integer optimization variables.

By solving the integer program (2.8), the optimal gear teeth choice emerged (GR = 23.36; sun gear: 12; planet gear: 16/53; ring gear: 81). It is worth noting that the planet gear teeth combination is chosen to be 16/53, which only shares the common factor of 1. Since the gear manufacturer cannot guarantee fixed teeth alignment between the two stages of the planet gear, the choice of planet gear teeth combination allows a tunable backlash with a

2.1° increment in the gearbox assembly. This design uses the principle of alignment, which is similar to the working principle of a Vernier caliper.

Gearbox Rotary Inertia

The rotary inertia of the gearbox is one of the key physical parameters of motor dynamics that need to be obtained to achieve better control performance. This section presents the derivation and the final expression for the lumped rotary inertia of the compound planetary gearbox design as shown in Fig. 2.4.

As a one DoF mechanical system, the gearbox inertia is derived using the Euler-Lagrangian equation. The mechanical energy consists only of the kinetic energy

$$\begin{aligned} KE_s &= \frac{1}{2} J_s \cdot \omega_s^2 \\ KE_c &= \frac{1}{2} J_c \cdot \omega_c^2 \\ KE_p &= \frac{1}{2} J_p \cdot \omega_p^2 + \frac{1}{2} m_p \cdot v_p^2, \end{aligned} \tag{2.9}$$

where KE denotes the kinetic energy; J represents the moment of inertia about the CoM; m_p is the mass of the planet gear; the linear velocity of the planet gear v_p is $v_p = \omega_p \cdot r_{p2}$. The Lagrangian of the gearbox is

$$\mathcal{L} = \mathcal{K} - \mathcal{P} = KE_s + KE_c + n_p \cdot KE_p, \tag{2.10}$$

where n_p is the number of planet gear. The equation of motion can be expressed in terms of the Lagrangian

$$T_s = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q}, \tag{2.11}$$

where T_s is the input torque to the sun gear; q is the sun gear rotation angle θ_s. The rotary inertia felt on the input side $J_{in} = T_s/\ddot{\theta}_s$, and the reflected inertia on the output side is

$J_{out} = GR^2 \cdot J_{in}$, where the expression of GR is in (2.6) and

$$J_{in} = J_s + \frac{n_p \cdot r_s^2(m_p \cdot r_{p2}^2 + J_p)}{(r_{p1} + r_{p2})^2} + \frac{J_c \cdot r_{p2}^2 \cdot r_s^2}{(r_{p1} + r_{p2})^2 \cdot r_c^2}. \tag{2.12}$$

The physical parameters of the gearbox are summarized in Table 2.2. The J_s, J_c, J_{out} for the ABAD (A), HIP (H), and KNEE (K) motor modules are different. The inertia parameters of each individual part are obtained from the CAD models, and then the lumped gearbox rotary inertia is calculated using (2.12) and (2.6).

Table 2.2: Physical Parameters of the Gearbox

Parameter	Value	Unit
r_s	2.40e-3	m
r_{p1}	1.06e-2	m
r_{p2}	3.20e-3	m
r_c	1.30e-2	m
n_p	3	N/A
m_p	4.94e-3	kg
J_p	2.63e-7	kg m^2
J_s	A: 1.17e-6; H: 1.81e-6; K: 1.81e-6	kg m^2
J_c	A: 7.43e-4; H: 1.87e-4; K: 5.86e-5	kg m^2
J_{out}	A: 1.40e-3; H: 1.19e-3; K: 1.06e-3	kg m^2

The inertia values as shown in Table 2.2 are used in Section 2.4.2 for swing leg control, and in Section 2.4.3 for contact detection.

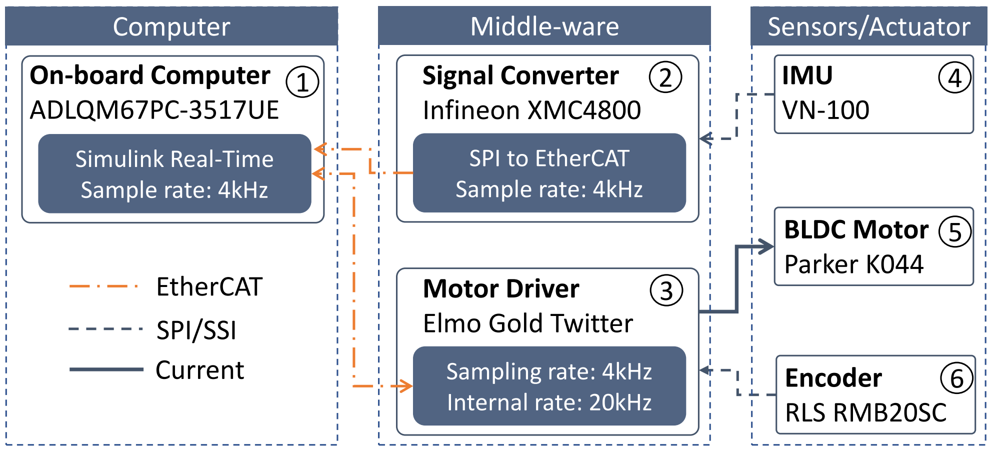


Figure 2.5: The electronic architecture for the quadrupedal robot *Panther*. The onboard SBC computer runs Simulink Real-Time and communicates via EtherCAT with a) Elmo^{TM} motor driver and b) Signal converter. The Elmo^{TM} motor driver controls torque commands to the BLDC motor and reads encoder readings. The signal converter consists of a custom PCB board and an Infineon^{TM} board, which collects IMU data.

2.3 Electronic System

The electronic system of *Panther* as shown in Fig. 2.5 centers around a single-board-computer (SBC) which runs the main control loop in Simulink Real-Time at 4kHz. The on-board computer PC104 with Intel i7-3517UE at 1.70 GHz communicates with the signal converter and motor drivers via EtherCAT at 4 kHz. The signal converter consists of a customized PCB and the Infineon^{TM} XMC4800 microcontroller and is used to collect the inertial measuring unit (IMU) readings in Serial Peripheral Interface (SPI) at 1 kHz. The Elmo^{TM} Gold Twitter motor drivers control the BLDC motors by running a torque control loop at 20 kHz. The 13 bit magnetic absolute encoder RLS-RMB20 is mounted on the input rotary shaft of each motor. Photos of the electronic parts are presented and in Fig. 2.6.

2.4 Software

This section presents some of the most important software modules of the quadrupedal robot control infrastructure. Section 2.4.1 introduces the state estimation using the complementary

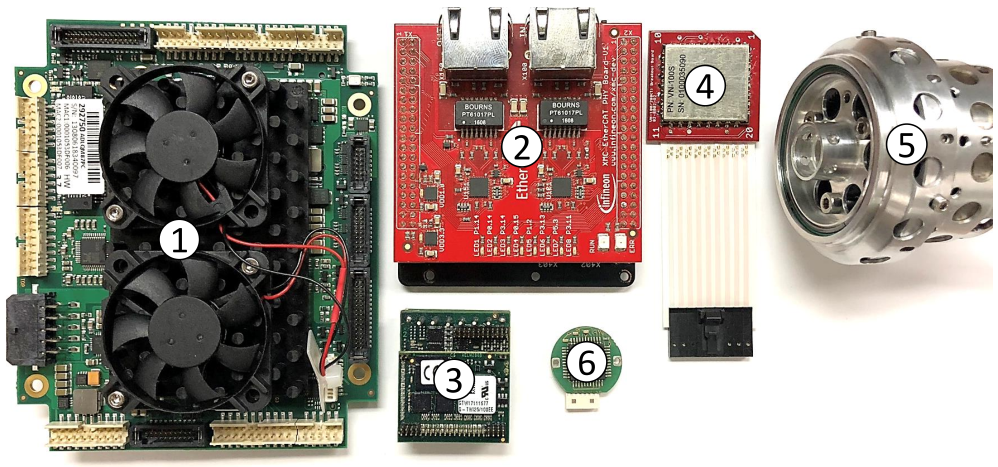


Figure 2.6: Photo of the electronic hardware components mentioned in Fig.2.5 (1) The on-board computer that runs SLRT (2) Infineon^{TM} XMC that interfaces with the IMU (3) The Elmo^{TM} Gold Twitter motor driver (4) The IMU unit (5) The Parker^{TM} K044 BLDC motor (6) The RLS^{TM} encoder

filter; Section 2.4.2 details the swing leg control; Section 2.4.3 presents the contact detection algorithm.

2.4.1 State Estimation

In the context of legged robot control, the goal of state estimation is to recover CoM position/velocity and torso orientation/angular velocity. The sensor readings from joint encoders and the IMU are used for the state estimation.

Kalman Filter [66] have been applied for a wide range of applications in legged robots. Meanwhile, simple linear single-input single-output (SISO) complementary filters [105] have been proven to work robustly in practice [21] [117]. The complementary filter performs low-pass filtering on a low-frequency estimation and high-pass filtering on a biased high-frequency estimation.

Considering the CoM velocity $\dot{\boldsymbol{p}}$ as an example. Based on the kinematic model shown in Fig. 2.7, a velocity estimate could be obtained from leg kinematics data $\dot{\boldsymbol{p}}^{enc}$. The accelerometer readings $\boldsymbol{a}^{acc}$ from the on-board IMU provides another sensor data. The CoM

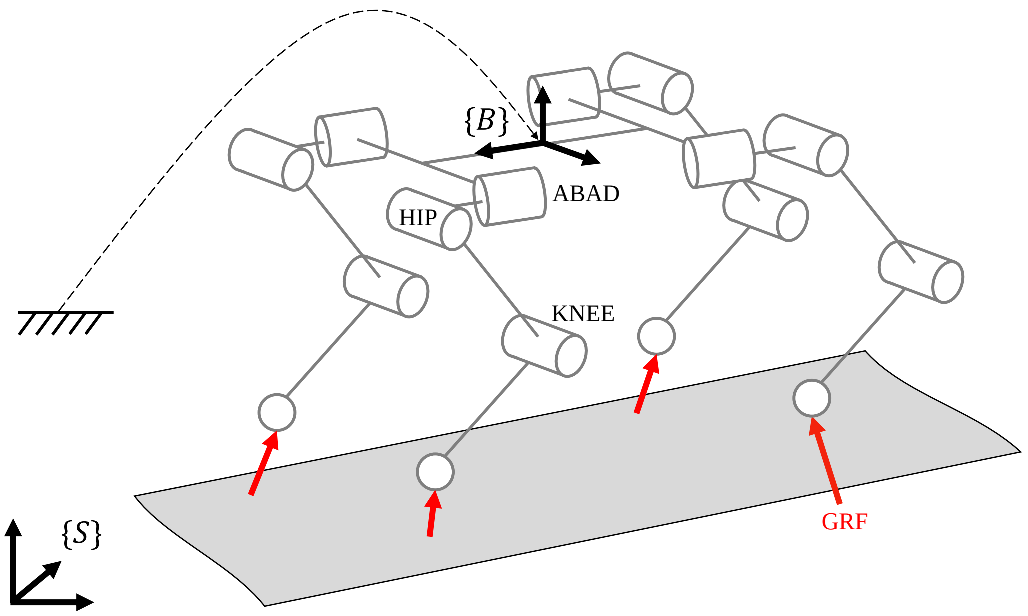


Figure 2.7: The kinematic structure of the *Panther* robot is a floating base system with actuated joints and unactuated base coordinates. The GRFs occur when a foot is in contact with the ground.

velocity estimate is obtained by combining both readings,

$$\begin{aligned} \dot{\boldsymbol{p}}_{k+1} &= \dot{\boldsymbol{p}}_k + \boldsymbol{a}_k \cdot \Delta t \\ \boldsymbol{a}_k &= \boldsymbol{a}_k^{acc} - \boldsymbol{K}_p^v(\dot{\boldsymbol{p}}_k - \dot{\boldsymbol{p}}_k^{enc}), \end{aligned} \tag{2.13}$$

where $\dot{\boldsymbol{p}}_k$ is the estimated velocity from the previous iteration; the subscript $(\cdot)_k$ is the discrete time index; Δt is the IMU sampling period; $\boldsymbol{K}_p^v$ is a tunable positive-definite diagonal gain matrix; $\boldsymbol{a}_k^{acc}$ is the accelerometer reading; $\dot{\boldsymbol{p}}_k^{enc}$ is the average of all the velocities from contact feet to CoM based on kinematic calculations. Similarly, the CoM position $\boldsymbol{p} \in \mathbb{R}^3$ is estimated by fusing the CoM position estimate from leg position kinematics $\boldsymbol{p}^{enc}$ and the estimated CoM velocity $\dot{\boldsymbol{p}}$. The torso orientation and angular velocity are directly measured using the IMU data.

2.4.2 Swing Leg Control

Although the motor and linkage inertia effect of the legs can be neglected during stance phases thanks to the light-weight design, it is considered in the swing leg controller for improved tracking performance. Each swing leg is modeled as a 3-link serial manipulator attached to the torso, which is considered as a stationary base. A schematics of the swing leg model is presented in Fig. 2.7. The swing leg controller includes both feed-forward and feedback terms, where the former is based on the workspace inverse dynamics control,

$$\boldsymbol{\tau}_{sw}^{ff} = \boldsymbol{D}(\boldsymbol{q})\boldsymbol{J}^{-1}(\boldsymbol{a}_x^f - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{2.14}$$

The feed-forward torque is denoted as $\boldsymbol{\tau}_{sw}^{ff}$; $\boldsymbol{D}(\boldsymbol{q})$ is the Inertia matrix and $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ includes the centrifugal, Corolis and gravitational terms of the swing leg; $\boldsymbol{q}, \dot{\boldsymbol{q}}$ are the joint angle and velocity vectors; $\boldsymbol{J}$ is the foot velocity Jacobian matrix and $\dot{\boldsymbol{J}}$ is its time derivative; $\boldsymbol{a}_x^f$ is workspace acceleration vector, which is defined as

$$\boldsymbol{a}_x^f = \ddot{\boldsymbol{p}}_d^f + \boldsymbol{K}_p^{ff}(\boldsymbol{p}_d^f - \boldsymbol{p}^f) + \boldsymbol{K}_d^{ff}(\dot{\boldsymbol{p}}_d^f - \dot{\boldsymbol{p}}^f), \tag{2.15}$$

where $\ddot{\boldsymbol{p}}_d^f$ is the desired foot workspace acceleration; $\boldsymbol{p}_d^f, \dot{\boldsymbol{p}}_d^f$ are the desired foot position and velocity; $\boldsymbol{K}_p^{ff}$, $\boldsymbol{K}_d^{ff}$ are the position and velocity gain matrices. The full swing leg controller consists of both feed-forward and feedback terms,

$$\boldsymbol{\tau}_{sw} = \boldsymbol{\tau}_{sw}^{ff} + \boldsymbol{K}_p^{fb}(\boldsymbol{p}_d^f - \boldsymbol{p}^f) + \boldsymbol{K}_d^{fb}(\dot{\boldsymbol{p}}_d^f - \dot{\boldsymbol{p}}^f), \tag{2.16}$$

where $\boldsymbol{K}_p^{fb}$ and $\boldsymbol{K}_d^{fb}$ are the position and velocity gain matrices for the feedback term of the swing leg controller.

The desired foot placement policy for *Panther* is a linear combination of a velocity-based feed-forward term and a capture-point [113] based feedback term.

$$\boldsymbol{p}^f_{step} = \boldsymbol{p}^h + \frac{T_{st}}{2}\dot{\boldsymbol{p}}^h_d + \sqrt{\frac{z^h_0}{g}}(\dot{\boldsymbol{p}}^h - \dot{\boldsymbol{p}}^h_d), \tag{2.17}$$

where $\boldsymbol{p}^f_{step}$ is the desired step location on the ground plane; $\boldsymbol{p}^h$ is the projection of the hip joint on the ground plane and $\dot{\boldsymbol{p}}^h$ is the corresponding velocity; $\dot{\boldsymbol{p}}^h_d$ is the desired hip velocity projected on the ground plane; T_{st} is the prescribed stance time; g is the gravitational acceleration constant; z^h_0 is the nominal hip height.

2.4.3 Contact Detection

Contact sensing plays a crucial role in legged locomotion. Proprioceptive sensing [142] is utilized in this work thanks to the highly-transparent actuation design. Specifically, the impact mitigation factor (IMF) [142] of the leg module is higher than other actuator designs that use harmonic gears or cycloidal gears. The contact detection algorithm uses the generalized momenta based disturbance observer [27], which only requires proprioceptive measurements $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and the commanded torque $\boldsymbol{\tau}$. Here, only the knee joints are considered in contact detection based on the assumption that the knee joint momentum is changed the most by the contact impact. The residual vector $\boldsymbol{r}_k$ quantifies the discrepancy between expected and measured knee joint momentum, and it is defined as,

$$\boldsymbol{r}_k = \boldsymbol{K}_I \cdot [\boldsymbol{I}^{kn}\dot{\boldsymbol{q}}^{kn}_k - \sum_{i=1}^{k}(\boldsymbol{\tau}^{kn}_i + \boldsymbol{r}_{i-1})\Delta t], \boldsymbol{r}_0 = 0, \tag{2.18}$$

where $\boldsymbol{r}_k \in \mathbb{R}^4$ is the residual vector for the four legs. k is the index for the current instance; $\boldsymbol{K}_I$ is a diagonal gain matrix; $\boldsymbol{I}^{kn}$ is the diagonal inertia matrix for all the knee joints; $\dot{\boldsymbol{q}}^{kn}_k$ is the vector of knee joint velocity; $\boldsymbol{\tau}^{kn}_k$ is the commanded knee torque; $\boldsymbol{r}_0$ is the initial value of the residual. The summation accumulates all the previous residuals and the commanded torque. Contact is declared when the residual vector $\boldsymbol{r}_k$ exceeds a threshold value r_{th}.

2.5 Implementation Details

This section presents the implementation details that are required for successful controller implementation on the robot hardware platform. The CoM position in the body frame (Section 2.5.1) and mass moment of inertia (Section 2.5.2) are measured experimentally. In addition, friction compensation (Section 2.5.3) and force calibration (Section 2.5.4) are performed for more accurate force control.

2.5.1 Center of Mass Location

The CoM location of a robot is usually obtained from the CAD model. However, for small robots, a large portion of the body mass is occupied by electronics, whose mass distribution cannot be exactly captured by the CAD model. Therefore, the CoM location of *Panther* is measured by suspending the robot by a string on multiple known locations. When the robot is stationary, the accelerometer reading is recorded. This procedure is repeated for several known attachment points on the robot. A bundle of lines can be constructed from the readings of the accelerometer and the position of the attachment points obtained from the CAD model. The CoM location can be obtained by solving a least-squares problem,

$$\underset{\boldsymbol{p}_{CoM}}{\operatorname{argmin}} \sum_i ||\boldsymbol{p}_{CoM} - \boldsymbol{l}_i||_2^2, \tag{2.19}$$

where $\boldsymbol{p}_{CoM}$ is the CoM location; $\{\boldsymbol{l}_i\}$ is the bundle of lines constructed from the accelerometer readings. The 2-norm takes the shortest Euclidean distance from $\boldsymbol{p}_{CoM}$ to the line. The legs are commanded to a stationary nominal position throughout the experiment.

2.5.2 Mass Moment of Inertia

Mass moment of inertia ${}^{B}\boldsymbol{I}$ is an important parameter for the dynamic modeling of the robot. However, the value directly obtained from the CAD model for a small robot may not be accurate due to the same reason for CoM location estimation in Section 2.5.1. Therefore, a linear version of the bifilar (two-wire) torsional pendulum [64] is used to obtain the mass moment of inertia.

2.5.3 Friction Compensation

The relatively high gear ratio of *Panther* induces higher joint friction, which is compensated for high-fidelity force control. Following [67], the friction is modeled as

$$\tau_f = c_1 \cdot \mathrm{sat}(\omega) + c_2 \cdot \tau_{motor} \cdot \mathrm{sat}(\omega), \tag{2.20}$$

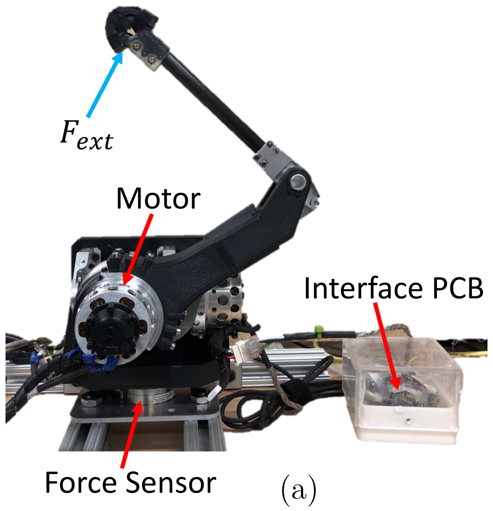
where ω is the output angular velocity; τ_{motor} is the commanded motor torque; c_1, c_2 are tunable constants that are motor-specific. τ_f is the friction compensation term and the output torque $\tau_{output} = \tau_{motor} + \tau_f$. The saturation function is defined as

$$\mathrm{sat}(\omega) = \begin{cases} -1 & \omega \leq -\omega_{thr} \\ 1/\omega_{thr} & -\omega_{thr} < \omega \leq \omega_{thr} \\ 1 & \omega_{thr} < \omega, \end{cases} \tag{2.21}$$

which serves as a relaxed version of the sign function. The threshold value ω_{thr} can be tuned to prevent chattering around the equilibrium point.

2.5.4 Force Calibration

In addition to friction, transmission backlash, nonlinear motor behavior and structural compliance also cause the end-effector forces to deviate from desired values. To compensate for these factors, a force calibration process is introduced. As shown in Fig. 2.8 (a), the robot leg is mounted on an ATI force-torque sensor to get the force measurement data as the ground truth. The robot leg is commanded to execute foot position control at a set-



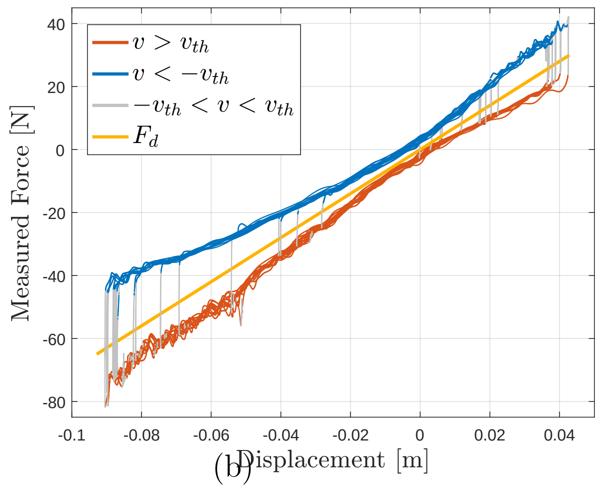


Figure 2.8: Experimental setup and result of force calibration. (a) The experiment is set up such that a leg module is rigidly mounted on a force sensor. (b) Measured vertical force versus displacement. The yellow line indicates the desired force; the red and blue points correspond to measured force with positive and negative velocities, respectively. The gray points correspond to the measured force when the velocity direction changes

point with high proportional gain (700 N/m). The experimenter applies external force $\boldsymbol{F}_{ext}$ to deviate the foot from the set-point, in a way such that the motions suffice to cover the operational range of forces and velocities during a jumping motion. Fig. 2.8 (b) presents the generated force error curve, where the yellow line is the desired force F_d, the orange curves correspond to the measured forces with positive velocity, and the blue curves for the negative velocity. The force error curve combined with the friction model (2.20) improves the force control capability of the robot.

2.6 Squat Jumping as Capability Test

The section presents the squat jumping experiment, which serves as a capability test on the robot *Panther*. The Quadratic Program-based controller introduced in Section 2.6.1 is implemented on the robot hardware for the squat jumping experiment. Section 2.6.2 presents the experiment data, which validates that *Panther* can perform highly dynamic maneuvers such as a 0.7 m high squat jump and land sequence[1].

2.6.1 Quadratic Program-based Controller

Squat jumping can be decomposed into three phases, namely, the jumping phase, the aerial phase, and the landing phase. During the jumping and landing phases, the robot is over-actuated since all four feet are in contact with the ground. Hence, there are infinitely many combinations of GRF that produce the desired net wrench $[\boldsymbol{f}_d^\top, \boldsymbol{\tau}_d^\top]^\top$, which is defined at the CoM. During the instance when the robot is over-actuated, a Quadratic Program (QP) is formulated and solved for the GRF while complying with the constraints such as the friction cone constraint.

$$\begin{aligned} &\text{minimize } \boldsymbol{e}_f^\top \boldsymbol{Q}_f \boldsymbol{e}_f + \boldsymbol{e}_\tau^\top \boldsymbol{Q}_\tau \boldsymbol{e}_\tau \\ &\text{subject to } \boldsymbol{f}_{min} \leq \boldsymbol{f}_i \leq \boldsymbol{f}_{max} \\ &\qquad\qquad \boldsymbol{f}_i \in \mathcal{K}, \forall i \in \{1,2,3,4\} \end{aligned} \tag{2.22}$$

where $\boldsymbol{e}_f = \boldsymbol{f} - \boldsymbol{f}_d$ and $\boldsymbol{e}_\tau = \boldsymbol{\tau} - \boldsymbol{\tau}_d$ are the error terms for total body wrench applied around the CoM, and $\boldsymbol{Q}_f, \boldsymbol{Q}_\tau \succeq 0$ are weighting matrices. The GRF $\boldsymbol{f}_i$ is bounded by $\boldsymbol{f}_{min}$ and $\boldsymbol{f}_{max}$. The friction cone constraint is approximated by the more conservative friction pyramid $\mathcal{K} := \{\boldsymbol{f}_i \in \mathbb{R}^3 | \ f_z \geq 0, ||f_{x/y}|| \leq \mu||f_z||\}$ to prevent the contact foot from slipping.

The desired body wrench is defined as the sum of feed-forward and feedback terms,

$$\begin{aligned} \boldsymbol{f}_d &= \boldsymbol{f}_{ff} - \boldsymbol{K}_p \boldsymbol{e}_p - \boldsymbol{K}_{\dot{p}} \boldsymbol{e}_{\dot{p}} \\ \boldsymbol{\tau}_d &= \boldsymbol{\tau}_{ff} - \boldsymbol{K}_R \boldsymbol{e}_R - \boldsymbol{K}_\omega \boldsymbol{e}_\omega \end{aligned} \tag{2.23}$$

where $\boldsymbol{f}_{ff}, \boldsymbol{\tau}_{ff}$ are the feed-forward wrench terms; $\boldsymbol{K}_p, \boldsymbol{K}_{\dot{p}}, \ \boldsymbol{K}_R, \boldsymbol{K}_\omega \succeq 0$ are the diagonal weighting matrices for CoM position, velocity, orientation and angular velocity, respectively. The feedback term of (2.23) is in proportional-derivative (PD) form, which mimics the

[1]Video clips of the squat jumping experiment is available here

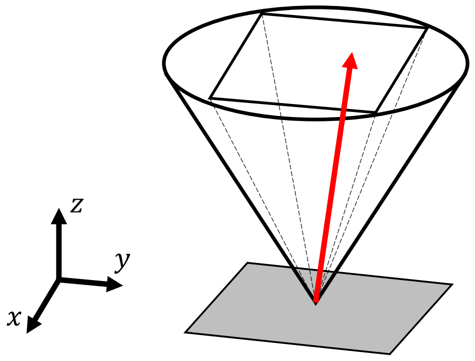


Figure 2.9: The friction cone constraint for the GRF is replaced by the more conservative friction pyramid constraint. The GRF shown as the red arrow is bounded within the friction pyramid to prevent slipping.

spring-damper behavior. The error terms in orientation and angular velocity are defined as in [80]

$$\begin{aligned} \boldsymbol{e}_R &= \frac{1}{2}(\boldsymbol{R}_d^\top \boldsymbol{R} - \boldsymbol{R}^\top \boldsymbol{R}_d)^\vee \\ \boldsymbol{e}_\omega &= {}^B\boldsymbol{\omega} - \boldsymbol{R}\boldsymbol{R}_d\ {}^B\boldsymbol{\omega}_d \end{aligned} \tag{2.24}$$

where $\boldsymbol{R}_d$, ${}^B\boldsymbol{\omega}_d$ are the desired orientation and angular velocity, and $\boldsymbol{R}$, ${}^B\boldsymbol{\omega}$ are the current measurement. The vee map is defined as $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$.

The feed-forward wrench trajectory in (2.23) is designed using off-line trajectory optimization (TO), where the desired state and control trajectories of a squat jumping motion are obtained. The robot is modeled as a point-mass moving in the vertical direction, since the torso is assumed to remain at the nominal orientation throughout the squat jump. The TO is formulated using the direct-collocation transcription method [136], and The solver *fmincon* is used to solve the NLP. The QP-based controller is expected to regulate the position and orientation deviation from the reference trajectory.

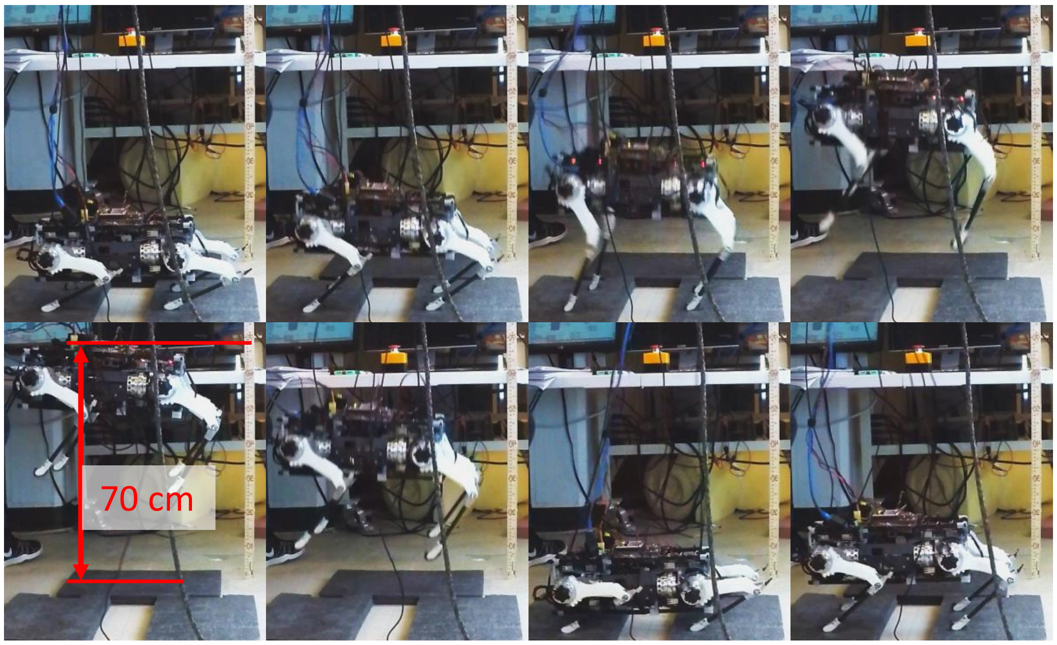


Figure 2.10: Sequential snapshots of the squat jumping experiment. The robot started from a static pose and achieved a maximum jumping height of 0.7 m before landing safely on the ground.

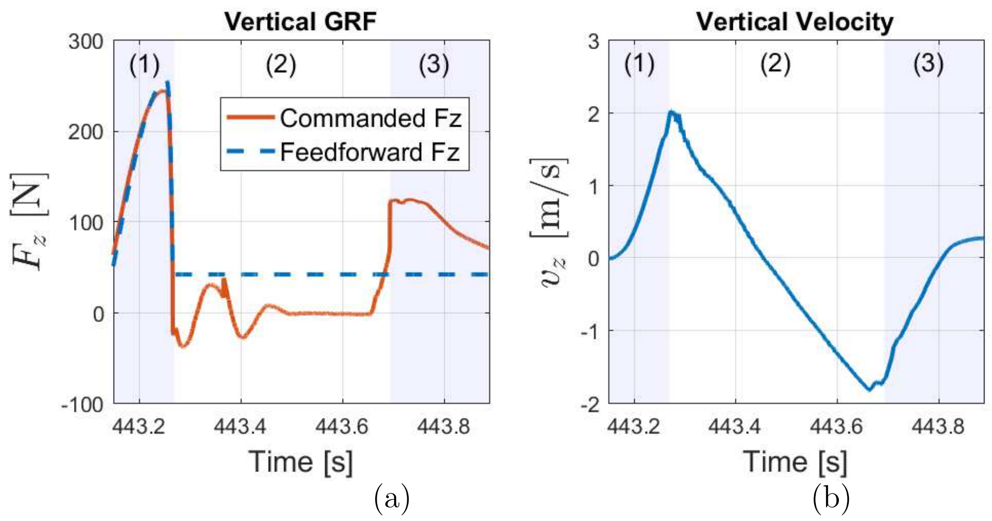
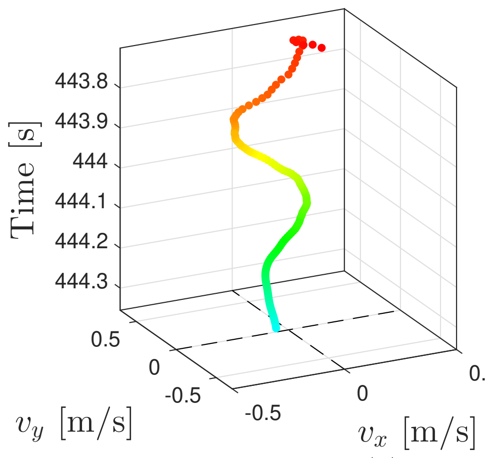
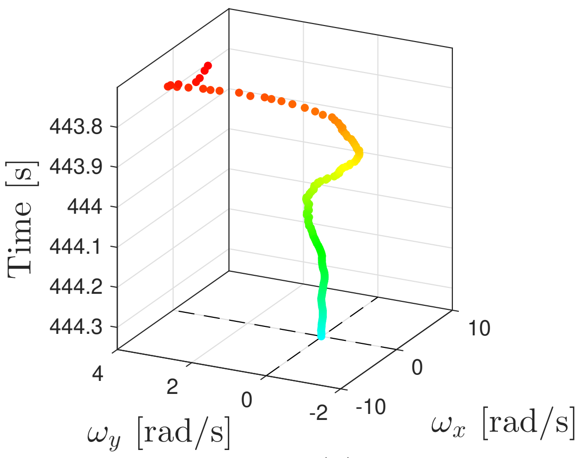


Figure 2.11: Experimental data of the squat jumping experiment. (a) commanded and desired vertical GRF (b) vertical CoM velocity. Shaded area (1) indicates the jumping phase; unshaded area (2) represents the aerial phase; shaded area (3) is the landing phase.



(a)



(b)

Figure 2.12: Stabilizing performance on the x-y plane during the landing phase. The completion percentile of landing is indicated by the color, where red indicates the beginning of landing and green the end. (a) The x-y liner velocities are stabilized (b) the x-y angular velocities are stabilized.

2.6.2 Experiment Results

The QP-based controller is implemented on *Panther*, and the squat jumping experiment is conducted. Sequential snapshots can be found in Fig. 2.10, where the robot reached a maximal jumping height of 0.7 m and landed on the ground safely.

As shown in Fig. 2.10, the robot started with a crouched position to maximize its jumping stroke, and then it applied a large jumping force while maintaining the body attitude. It is crucial to regulate the angular velocity to zero since residual angular velocity at take-off will result in a tilted landing pose due to the extended aerial time. During the aerial phase, the swing leg controller described in Section 2.4.2 tracks the desired foot trajectory. After reaching a maximum height of 0.7 m, the robot fell down and detected the touchdown using the contact detection algorithm (Section 2.4.3) and initiated the landing phase. The torso of the robot is stabilized by solving QP problems at 4 kHz using the solver qpOASES [42] in the on-board computer. The QP solver qpOASES is chosen here because it uses the active-set method [145], and is efficient in solving QP with a small number of decision variables.

Fig. 2.11 (a) shows the feed-forward and commanded vertical GRF in the squat jumping experiment, and Fig. 2.11 (b) presents the vertical velocity of the CoM. During the jumping phase (1), commanded GRF closely follow the feed-forward value and the velocity of the robot rapidly accelerates, as shown in Fig. 2.11 (b). During the aerial phase (2), the robot experiences free fall. The oscillatory force in aerial phase (2) is due to the inertia effect of the swing leg tracking the set-point. During the landing phase (3), the robot handles the impact and decelerates the body in a compliant manner. Fig. 2.12 demonstrates the capability of the QP-based controller to regulate horizontal motion. Fig. 2.12 (a) shows that the x-y linear velocities are controlled to zero, and Fig. 2.12 (a) shows the roll (x) and pitch (y) angular velocities are modulated to zero at the end of the landing phase.

The lessons learned from this experiment are that the landing phase is more difficult to stabilize compared with jumping phase. That is because the robot has to handle the landing impact and regulate the state, which deviates from the desired state due to the extended aerial phase. Another lesson learned is that high control frequency is crucial in the landing phase because it allows high task-space damping gain $\boldsymbol{K}_{\dot{p}} = diag(30, 30, 50)[Ns/m]$, which is responsible for absorbing the impulse upon impact and dissipate the kinetic energy. Due to the small size of decision variable (12 variables), the QPs can be solved by the active-set method in qpOASES efficiently. It might be counter-intuitive that too high of a proportional gain may destabilize landing since the robot tends to bounce up and lose contact with the ground.

2.7 Summary

This chapter presents the design synthesis of a small and agile quadruped robot -*Panther*. An overview of the robot platform is presented and the leg module design features are described. Details about the actuator design in Section 2.2 can be considered as complementary material to our design paper [34]. In addition, the electronic system, software framework, and implementation details are exhibited. The capability of the *Panther* robot is showcased in a squat jumping experiment, where a maximal jumping height of 0.7 m is reached before the robot landed safely.

Chapter 3

Representation-Free Model Predictive Control

This chapter presents a model predictive control (MPC) framework for controlling various dynamic motions of *Panther* in 3D space. This formulation directly represents the rotational dynamics using the rotation matrix, which eliminates the issues associated with the use of Euler angles and quaternions as the orientation representations. With a variation-based linearization scheme and a carefully constructed cost function, the MPC control law is transcribed to a standard quadratic program (QP). This representation-free MPC (RF-MPC) can stabilize dynamic motions with large orientation excursion using a single controller, which was previously achieved by switching among candidate controllers. RF-MPC operated at a real-time rate of 250 Hz on *Panther*, enabling experimental results including periodic gaits and a controlled tumble, which involves singularity in the 3D maneuvers[1] [2]. This content of this chapter is based on our work in [33] and [32].

3.1 Introduction

The ability to represent the orientation of the object of interest is one of the fundamental elements in robotics research, whether it be manipulation, vision, navigation, or locomotion. The orientation of a single rigid body in 3D space is defined by a rotation matrix $\boldsymbol{R} \in SO(3)$, where $SO(3)$ is the special orthogonal group. Due to the complex manifold dynamics associated with the rotation matrix, orientation representations such as Euler angles and quaternion [22] are more widely adopted in the legged robot community. This

[1]Video clips of the simulation and experiment is available here

[2]Simulation code is open-sourced and available in the GitHub repo here

section investigates these two commonly-used orientation representations, and motivates the proposition of the representation-free framework, which directly use the rotation matrix for orientation.

3.1.1 Euler Angles

Euler angles consist of three consecutive rotation angles along three mutually orthogonal axes for orientation representation. Due to its intuitive definition and ease of visualization, the Euler angle representation is frequently used in robotics applications. However, the singularity issue [128] (also known as Gimbal lock) inherent to Euler angles prohibits the motion design from passing the singular configurations, which are commonly placed at pitch angle $= \pm\frac{\pi}{2}$. Although this defect of Euler angles will not affect locomotion tasks that do not involve large orientation excursion, it restricts the quadrupedal robot from executing motions such as climbing up trees or walls. Furthermore, this issue manifests itself within the "vicinity" of singularity and can destabilize the system. A simple pose control simulation is performed to illustrate this point. Specifically, two MPC controllers are implemented and benchmarked in a pose control task with (a) the proposed representation-free model predictive control (RF-MPC) and (b) an MPC that uses Euler angle for orientation representation (EA-MPC). The EA-MPC is implemented based on the convex MPC [29] with parameters from [94].

Assuming EA-MPC adopts the Z-Y-X sequence in body frame $\{B\}$, which is equivalent to the X-Y-Z sequence in stationary inertial frame $\{S\}$. The Euler angles $\mathbf{\Theta} = [\phi\ \theta\ \psi]^\top$, where ϕ is the roll, θ is the pitch, and ψ is the yaw. The attitude of frame $\{B\}$ is expressed by a sequence of rotations in frame $\{S\}$ as

$$\boldsymbol{R} = \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi), \tag{3.1}$$

where $\boldsymbol{R}_x(\phi)$ means a positive rotation of angle ϕ around the x-axis of frame $\{S\}$.

We define $\mathcal{T}_{\boldsymbol{\Theta}} : \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times 3}$ to be the matrix that converts $\dot{\boldsymbol{\Theta}}$ to the angular velocity expressed in $\{S\}$ as

$$\boldsymbol{\omega} = \mathcal{T}_{\boldsymbol{\Theta}} \cdot \dot{\boldsymbol{\Theta}} = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\sin(\psi) & 0 \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 \\ -\sin(\theta) & 0 & 1 \end{bmatrix} \dot{\boldsymbol{\Theta}}. \tag{3.2}$$

The matrix $\mathcal{T}_{\boldsymbol{\Theta}}$ in equation (3.2) is invertible when $\theta \neq \pm\frac{\pi}{2}$, and $\dot{\boldsymbol{\Theta}}$ can be calculated using the following equation

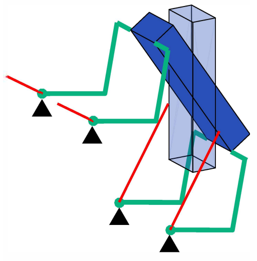
$$\dot{\boldsymbol{\Theta}} = \begin{bmatrix} \cos(\psi)/\cos(\theta) & \sin(\psi)/\cos(\theta) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ \cos(\psi)\tan(\theta) & \sin(\psi)\tan(\theta) & 1 \end{bmatrix} \boldsymbol{\omega}. \tag{3.3}$$

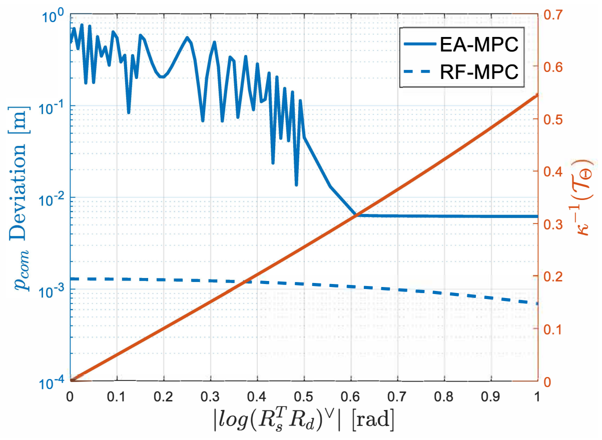
In this work, we use the following metric

$$\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}}) \in (0, 1] \tag{3.4}$$

to quantify the distance to the singularity of Euler angles, where $\kappa(\cdot)$ calculates the condition number of a matrix. When the robot approaches singular poses, the condition number $\kappa(\mathcal{T}_{\boldsymbol{\Theta}})$ increases rapidly, and its inverse $\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}})$ tends to 0.

Here, a pose control simulation is conducted to investigate the singularity of Euler angles. As shown in Fig. 3.1(a), the singular pose $\boldsymbol{R}_s$ is shown as the shadowed box; the desired pose $\boldsymbol{R}_d$ is shown as the solid box. All feet of the robot are assumed to be fixed in this simulation so that the GRF can be in any direction. The desired poses are varied from the singular pose to the pose rotated 1 rad around the +y axis. A 0.5 s simulation is conducted in each desired pose and the CoM deviation at the end of the simulation is plotted for both RF-MPC and EA-MPC. As can be observed in Fig. 3.1(b), while RF-MPC remains stable, EA-MPC is significantly affected by singularity once $|log(\boldsymbol{R}_s^\top \boldsymbol{R}_d)^\vee| < 0.3$ rad, which corresponds to




EA-MPC
RF-MPC
p_{com} Deviation [m]
$\kappa^{-1}(\mathcal{T}_\Theta)$
$|log(R_s^T R_d)^\vee|$ [rad]

(a) (b)

Figure 3.1: The pose control simulation result of a investigation on the singularity of Euler angles. (a) The schematics of the pose control, where the shaded box represents the singular pose; the solid box represents the commanded pose; the red lines represent the GRF. (b) The CoM position deviations (log scale) after a 0.5 s simulation of both RF-MPC and EA-MPC are plotted against $|log(\boldsymbol{R}_s^\top \boldsymbol{R}_d)^\vee|$. The metric for distance from singularity κ^{-1} is the red line.

$\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}}) < 0.15$.

3.1.2 Quaternion

Quaternion [129] is a singularity-free orientation representation. However, as mentioned in [8], quaternions have two local charts that cover the special orthogonal group $SO(3)$ twice. This ambiguity can cause the unwinding phenomenon [8], where the body may start arbitrarily close to the desired attitude and yet rotate large angles before reaching the desired orientation. Widely adopted by the unmanned aerial vehicle (UAV) community, quaternions are often used in reactive controllers which instantaneously respond to the state of the vehicle. Sign function has been used in the reactive controller [152] to eliminate the ambiguity of the quaternion representation. In [89], hybrid dynamic algorithm has been introduced to solve the ambiguity of the quaternion representation. However, for predictive controllers such as model predictive control (MPC), switching local charts is undesirable.

The orientation of a rigid body is originally parameterized by the rotation matrix, which evolves on $SO(3)$ [17]. Although other orientation representations can be re-aligned to avoid their corresponding issues in a specific motion, the rotation matrix possesses advantages as global parametrization that is compact and singularity-free. However, an extensive formulation is required, as is introduced in Section 3.2.

3.2 Representation-Free MPC Formulation

Model Predictive Control (MPC), also known as Receding Horizon Control (RHC), considers a model of the system to be controlled and repeatedly solves for the optimal control input subject to the state and control constraints. At each sampling time, a finite horizon optimal control problem is solved and the control signal for the first time-step is applied to the system during the following sampling interval. After that, the same process is repeated with the updated measurements. MPC-based controllers have the capability to incorporate various constraints that are essential to legged locomotion, including unilateral ground reaction force (GRF) and friction cone constraints. Besides, MPC can provide control laws that are discontinuous [91], which can not be easily achieved by conventional control techniques.

The MPC control law can be obtained by solving the following constrained optimization problem

$$\begin{aligned}
\text{minimize} \quad & \ell_T(\boldsymbol{x}_{t+N|t}) + \sum_{k=0}^{N-1} \ell(\boldsymbol{x}_{t+k|t}, \boldsymbol{u}_{t+k|t}) && (3.5a)\\
\text{subject to} \quad & \boldsymbol{x}_{t+k+1|t} = \boldsymbol{f}(\boldsymbol{x}_{t+k|t}) + \boldsymbol{g}(\boldsymbol{x}_{t+k|t})\boldsymbol{u}_{t+k|t} && (3.5b)\\
& \qquad k = 0, \cdots, N-1 && (3.5c)\\
& \boldsymbol{x}_{t+k|t} \in \mathbb{X}, k = 0, \cdots, N-1 && (3.5d)\\
& \boldsymbol{u}_{t+k|t} \in \mathbb{U}, k = 0, \cdots, N-1 && (3.5e)\\
& \boldsymbol{x}_{t|t} = \boldsymbol{x}(t) = \boldsymbol{x}_{op} && (3.5f)\\
& \boldsymbol{x}_{t+N|t} \in \mathbb{X}_f && (3.5g)
\end{aligned}$$

where $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^m$ are the state and input vectors, respectively; $\ell_T : \mathbb{R}^n \to \mathbb{R}$ is the terminal cost function; $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the stage cost function; N is the prediction horizon; $\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}$ is the control affine dynamic update equation; $\mathbb{X} \subseteq \mathbb{R}^n, \mathbb{U} \subseteq \mathbb{R}^m$ are the feasible polyhedral sets for the state and control; $\mathbb{X}_f$ is the final state set; $\boldsymbol{x}_{t+k|t}$ denotes the state vector at time $t + k$ predicted at time t, using the current state measurement $\boldsymbol{x}_{t|t} = \boldsymbol{x}_{op}$, where the subscript $(\cdot)_{op}$ denotes the variables at the current operating point. The operating point in this manuscript is defined as the current state $\boldsymbol{x}_{op}$ and control $\boldsymbol{u}_{op}$.

In the case that the dynamic update equation $\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}$ is a nonlinear function, a nonlinear MPC (NMPC) can be formulated and solved as a general nonlinear program (NLP) by utilizing trajectory optimization (TO) techniques such as multiple shooting [13] or direct collocation [136].

Our main objective is to formulate a real-time executable MPC scheme for controlling quadruped robotics to perform a variety of dynamic motions. To meet the real-time requirement, the optimization problem posed by the MPC has to be solved robustly at a high rate on the mobile embedded computer, which has limited computational resources. Hence, a simplified model is adopted to reduce the dimensionality of the optimization problem. Since the mass of all legs combined is less than 10% of the total body mass, a single rigid body model serves as a reasonable approximation.

3.2.1 3D Single Rigid Body Model

To mitigate the issue of demanding computational requirement of MPC for high Degrees of Freedom (DoF) system models, simple models or templates [43] that capture the dominant system dynamics are used instead. Templates such as the Linear Inverted Pendulum [65] (LIP) is widely used in humanoid robots [114, 149, 151]. Centroidal dynamics [103] model is used in [26] [140] [81] to capture the major dynamic effect of the complex full-body dynamics model. The quadrupedal robot community has seen an increasing number of work that utilizes the SRB model in three-dimensional (3D) space, which assumes that the entire

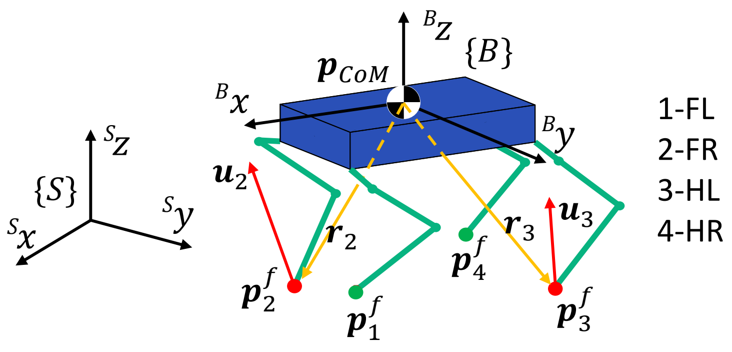


Figure 3.2: Illustration of coordinate systems and the 3D single rigid-body model. $\{S\}$ is the inertia frame and $\{B\}$ is the body attached frame. $\boldsymbol{r}_i$ is the position vector from CoM to each foot in $\{S\}$ and $\boldsymbol{u}_i$ is the GRF of i^{th} contact foot expressed in $\{S\}$. The convention for the numbering of feet is such that FL stands for front-left leg, and HR stands for the hind-right leg.

mass of the robot is lumped into a single rigid body (SRB). The simplicity of the SRB model is enabled by the light leg design, whose inertial effect is negligible compared with the body. Let the state of the single rigid body model be

$$\boldsymbol{x} := [\boldsymbol{p} \;\; \dot{\boldsymbol{p}} \;\; \boldsymbol{R} \;\; {}^{B}\boldsymbol{\omega}] \in \mathbb{R}^{18}, \tag{3.6}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position of the body Center of Mass (CoM); $\dot{\boldsymbol{p}} \in \mathbb{R}^3$ is the CoM velocity; $\boldsymbol{R} \in SO(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times3} | \boldsymbol{R}^\top \boldsymbol{R} = \mathbb{I}, \det(\boldsymbol{R}) = +1\}$ is the rotation matrix of the body frame $\{B\}$ expressed in the inertial frame $\{S\}$; $\det(\cdot)$ calculates the determinant of a matrix and $\mathbb{I}$ is the 3-by-3 identity matrix. Here, the rotation matrix $\boldsymbol{R}$ is reshaped into vector form. ${}^{B}\boldsymbol{\omega} \in \mathbb{R}^3$ indicates the angular velocity vector expressed in the body frame $\{B\}$. Variables without superscript on the upper-left corner can be assumed to be expressed in the inertial frame. The illustration of the coordinate system can be found in Fig. 3.2.

The input to the dynamical system is the GRF $\boldsymbol{u}_i \in \mathbb{R}^3$ at contact foot locations $\boldsymbol{p}_i^f \in \mathbb{R}^3$. The GRFs create the external wrench to the rigid body, where $i \in \{1, 2, 3, 4\}$ is the index for the front left (FL), front right (FR), hind left (HL) and hind right (HR), respectively, as shown in Fig. 3.2. The foot positions $\boldsymbol{p}_i^f$ relative to CoM are denoted as $\boldsymbol{r}_i = \boldsymbol{p}_i^f - \boldsymbol{p}$.

Therefore, the net external wrench $\boldsymbol{\mathcal{F}} \in \mathbb{R}^6$ exerted on the body is:

$$\boldsymbol{\mathcal{F}} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{\tau} \end{bmatrix} = \sum_{i=1}^{4} \begin{bmatrix} \mathbb{I} \\ \hat{\boldsymbol{r}}_i \end{bmatrix} \boldsymbol{u}_i, \tag{3.7}$$

where $\boldsymbol{F}$ and $\boldsymbol{\tau}$ are the total force and torque applied at the CoM; the hat map $\hat{(\cdot)} : \mathbb{R}^3 \to \mathfrak{so}(3)$ maps an element from $\mathbb{R}^3$ to the space of skew-symmetric matrices $\mathfrak{so}(3)$, which represents the cross-product under multiplication as $\hat{\boldsymbol{\alpha}}\boldsymbol{\beta} = \boldsymbol{\alpha} \times \boldsymbol{\beta}$, for all $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^3$. The inverse of the hat map is the vee map $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$. The full dynamics of the rigid body can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \ddot{\boldsymbol{p}} \\ \dot{\boldsymbol{R}} \\ {}^B\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \frac{1}{M}\boldsymbol{F} + \boldsymbol{a}_g \\ \boldsymbol{R} \cdot {}^B\hat{\boldsymbol{\omega}} \\ {}^B\boldsymbol{I}^{-1}(\boldsymbol{R}^\top \boldsymbol{\tau} - {}^B\hat{\boldsymbol{\omega}}{}^B\boldsymbol{I}{}^B\boldsymbol{\omega}) \end{bmatrix}, \tag{3.8}$$

where $\boldsymbol{u} = [\boldsymbol{u}_1^\top, \boldsymbol{u}_2^\top, \boldsymbol{u}_3^\top, \boldsymbol{u}_4^\top]^\top \in \mathbb{R}^{12}$ is the control vector; M is the mass of the rigid body; $\boldsymbol{a}_g = [0, 0, -g]^\top$ is the gravitational acceleration vector; ${}^B\boldsymbol{I} \in \mathbb{R}^{3\times 3}$ is the fixed moment of inertia tensor in the body frame $\{B\}$. The inertia properties of the robot can be found in Table 2.1.

To develop a representation-free control approach, we decided to directly parameterize orientation using the rotation matrix. This completely avoids the singularities and complexities when using local coordinates such as Euler angles. It also avoids ambiguities when using quaternions to represent attitude dynamics. As the quaternion double covers the special orthogonal group $SO(3)$, the control design needs to switch between the local charts.

The rotational dynamics of (3.8) is nonlinear for it involves the rotation matrix $\boldsymbol{R}$, which evolves on the $SO(3)$ manifold. In Section 3.2.2 we present a variation-based linearization scheme for linearizing the rotational dynamics.

3.2.2 Variation-based Linearization

Although the nonlinear MPC (3.5) can be solved to obtain the control input, the presence of local optimum resulting from the nonlinear dynamics complicates the solution process. Furthermore, convoluted nonlinear optimization does not lend itself well to embedded implementations. To meet the real-time constraint, we strive to formulate the MPC as a Quadratic Program (QP) that can be efficiently solved on embedded systems. A variation-based linearization scheme for the rotation matrix is proposed to linearize the nonlinear rotational dynamics. The error on the non-Euclidean $SO(3)$ manifold is approximated by the corresponding variation [85] with respect to the operating point. Then the variational dynamics is derived based on the system model (3.8) in the manner of [146]. Recent work [20] achieved underactuated two-leg balancing on MIT Mini Cheetah using variational-based linearization on the $SO(3)$ manifold.

Assuming that the predicted variables are close to the operating point, the variation of the rotation matrix on $\mathfrak{so}(3)$ can be approximated by $\delta\boldsymbol{R}$ using the derivative of the error function on $SO(3)$ as in [80]. The variation $\delta\boldsymbol{R} \in \mathfrak{so}(3)$ is a local approximation of the displacement between two points on the $SO(3)$ manifold. The rotation matrix at the k^{th} prediction step is approximated using the first-order Taylor expansion of matrix exponential map,

$$\boldsymbol{R}_k \approx \boldsymbol{R}_{op}\exp(\delta\boldsymbol{R}_k) \approx \boldsymbol{R}_{op}(\mathbb{I} + \delta\boldsymbol{R}_k), \tag{3.9}$$

where we use the commutativity of small rotations based on the assumption of $\delta\boldsymbol{R}$ being small.

The nonlinear dynamics of the rotation matrix is given as

$$\dot{\boldsymbol{R}} = \boldsymbol{R}^B\hat{\boldsymbol{\omega}}, \tag{3.10}$$

where the first-order approximation of rotation matrix $\boldsymbol{R}_k$ is presented in (3.9). To get a linear approximation for $\dot{\boldsymbol{R}}$, we define the variation of angular velocity $\delta\boldsymbol{\omega}_k$ as

$$\delta\boldsymbol{\omega}_k = \boldsymbol{\omega}_k - \boldsymbol{R}_k^\top \boldsymbol{R}_{op}\boldsymbol{\omega}_{op}, \tag{3.11}$$

where the transport map $\boldsymbol{\omega}_{op} \rightarrow \boldsymbol{R}_k^\top \boldsymbol{R}_{op}\boldsymbol{\omega}_{op}$ enables comparison between tangent vectors at different points. This procedure is required because the tangent vectors $\dot{\boldsymbol{R}}_k \in T_{R_k}SO(3)$ and $\dot{\boldsymbol{R}}_{op} \in T_{R_{op}}SO(3)$ lie in different tangent spaces and cannot be compared directly, where $T_{R_{op}}SO(3)$ refers to the tangent space of $SO(3)$ at $\boldsymbol{R}_{op}$. Hence, the angular velocity $\boldsymbol{\omega}$ can be deducted from (3.11) as

$$\begin{aligned}\boldsymbol{\omega}_k =& \boldsymbol{R}_k^\top \boldsymbol{R}_{op}\boldsymbol{\omega}_{op} + \delta\boldsymbol{\omega}_k \\ =& (\mathbb{I} + \delta\boldsymbol{R}_k)^\top \boldsymbol{\omega}_{op} + \delta\boldsymbol{\omega}_k \\ =& \boldsymbol{\omega}_{op} + \delta\boldsymbol{\omega}_k - \delta\boldsymbol{R}_k\boldsymbol{\omega}_{op},\end{aligned} \tag{3.12}$$

where $\boldsymbol{R}_k$ is replaced by the expression in (3.9). The last step in (3.12) is due to the fact that $\delta\boldsymbol{R}_k$ is a skew-symmetric matrix by construction. Applying the hat map $\hat{(\cdot)}$ to $\boldsymbol{\omega}_k$ and substituting (3.12) into (3.10) yields

$$\begin{aligned}\dot{\boldsymbol{R}}_k &= \boldsymbol{R}_k\hat{\boldsymbol{\omega}}_k = \boldsymbol{R}_{op}(\mathbb{I} + \delta\boldsymbol{R}_k)(\hat{\boldsymbol{\omega}}_{op} + \widehat{\delta\boldsymbol{\omega}_k} - \delta\widehat{\boldsymbol{R}_k\boldsymbol{\omega}_{op}}) \\ &= \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op} + \boldsymbol{R}_{op}\widehat{\delta\boldsymbol{\omega}_k} - \boldsymbol{R}_{op}\delta\widehat{\boldsymbol{R}_k\boldsymbol{\omega}_{op}} + \boldsymbol{R}_{op}\delta\boldsymbol{R}_k\hat{\boldsymbol{\omega}}_{op},\end{aligned} \tag{3.13}$$

where the higher order variation terms are neglected. Using the properties of cross product in [40] Table 2.1, the following equality

$$\delta\widehat{\boldsymbol{R}_k\boldsymbol{\omega}_{op}} = \delta\boldsymbol{R}_k\hat{\boldsymbol{\omega}}_{op} - \hat{\boldsymbol{\omega}}_{op}\delta\boldsymbol{R}_k, \tag{3.14}$$

is used to derive the expression of $\dot{\boldsymbol{R}}_k$

$$\dot{\boldsymbol{R}}_k = \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op} + \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op}\delta\boldsymbol{R}_k + \boldsymbol{R}_{op}\widehat{\delta\boldsymbol{\omega}_k}. \tag{3.15}$$

The dynamics of angular velocity $\dot{\boldsymbol{\omega}}_k$ is linearized as

$$\begin{aligned}{}^B\boldsymbol{I}\dot{\boldsymbol{\omega}}_k =& \boldsymbol{R}_{op}^\top \boldsymbol{\tau}_{op} + \delta \boldsymbol{R}_k^\top \boldsymbol{\tau}_{op} + \boldsymbol{R}_{op}^\top \delta\boldsymbol{\tau}_k + \\ & - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I}\boldsymbol{\omega}_{op} - \delta\hat{\boldsymbol{\omega}}_k{}^B\boldsymbol{I}\boldsymbol{\omega}_{op} - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I}\delta\boldsymbol{\omega}_k,\end{aligned} \tag{3.16}$$

in which $\delta\boldsymbol{\tau}_k$ is the variation of the net torque,

$$\delta\boldsymbol{\tau}_k = (\sum_{i=1}^{4} \hat{\boldsymbol{u}}_{i,op})\delta\boldsymbol{p}_k + (\sum_{i=1}^{4} \hat{\boldsymbol{r}}_{i,op} \cdot \delta\boldsymbol{u}_{i,k}), \tag{3.17}$$

where $\boldsymbol{u}_{op}$ is GRF applied at the current step, $\delta\boldsymbol{u}$ is the variation of GRF from $\boldsymbol{u}_{op}$. Note that the GRF applied at the next time step is $\boldsymbol{u}_{op} + \delta\boldsymbol{u}_1$.

The linearized dynamics will be used as affine dynamics as well as in the construction of objective function in the MPC formulation.

3.2.3 Vectorization

After the dynamics of rotation matrix $\boldsymbol{R}$ and angular velocity $\boldsymbol{\omega}$ are linearized, the matrix variables in (3.15) and (3.16) are still difficult to be formulated into the standard QP form. This section proposes a vectorization technique that uses the Kronecker product [45] to transform matrix-matrix products into matrix-vector products.

Let us define a vector $\boldsymbol{\xi} \in \mathbb{R}^3$ be such that the skew-symmetric matrix $\hat{\boldsymbol{\xi}} = \delta\boldsymbol{R} \in \mathfrak{so}(3)$ is an element in the tangent space at the operating point. Let $\boldsymbol{N} \in \mathbb{R}^{9\times3}$ be a constant matrix such that $vec(\hat{v}) = \boldsymbol{N}v, \forall v \in \mathbb{R}^3$, where $vec(\cdot)$ is the vectorization function. Then $vec(\delta\boldsymbol{R}) = \boldsymbol{N} \cdot \boldsymbol{\xi}$. The second and third terms of (3.15) can be vectorized as:

$$\begin{aligned} vec(\boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op}\delta\boldsymbol{R}_k) &= (\mathbb{I} \otimes \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op})\boldsymbol{N}\boldsymbol{\xi}_k \\ vec(\boldsymbol{R}_{op}\widehat{\delta\boldsymbol{\omega}_k}) &= (\mathbb{I} \otimes \boldsymbol{R}_{op})vec(\widehat{\delta\boldsymbol{\omega}_k}), \end{aligned} \tag{3.18}$$

where $\otimes$ is the Kronecker tensor operator. To derive the expression for $vec(\widehat{\delta\boldsymbol{\omega}_k})$, one plugs

(3.9) into (3.11)

$$vec(\widehat{\delta\boldsymbol{\omega}_k}) = \boldsymbol{N}(\boldsymbol{\omega}_k - \boldsymbol{\omega}_{op} + \hat{\boldsymbol{\omega}}_{op}\boldsymbol{\xi}_k). \tag{3.19}$$

The vectorized version of (3.15) is

$$vec(\dot{\boldsymbol{R}}_k) = \boldsymbol{C}_{\xi}^{c} + \boldsymbol{C}_{\xi}^{\xi}\boldsymbol{\xi}_k + \boldsymbol{C}_{\xi}^{\omega}\boldsymbol{\omega}_k, \tag{3.20}$$

where the constants are defined as

$$\begin{aligned}\boldsymbol{C}_{\xi}^{c} &= vec(\boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op}) - (\mathbb{I}\otimes\boldsymbol{R}_{op})\boldsymbol{N}\boldsymbol{\omega}_{op}\\ \boldsymbol{C}_{\xi}^{\xi} &= (\mathbb{I}\otimes\boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op})\boldsymbol{N} - (\mathbb{I}\otimes\boldsymbol{R}_{op})\boldsymbol{N}\hat{\boldsymbol{\omega}}_{op}\\ \boldsymbol{C}_{\xi}^{\omega} &= (\mathbb{I}\otimes\boldsymbol{R}_{op})\boldsymbol{N}.\end{aligned} \tag{3.21}$$

The discrete orientation dynamics in terms of $\boldsymbol{\xi}$ is derived by propagating the rotation matrix using the forward Euler integration scheme,

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k + \dot{\boldsymbol{R}}_k dt, \tag{3.22}$$

where dt is the MPC sampling time. When the rotation matrix is approximated by the first-order expansion in (3.9), the above expression can be simplified to the following form,

$$\delta\boldsymbol{R}_{k+1} = \delta\boldsymbol{R}_k + \boldsymbol{R}_{op}^{\top}\dot{\boldsymbol{R}}_k dt. \tag{3.23}$$

Vectorizing (3.23) gives

$$\boldsymbol{N}\boldsymbol{\xi}_{k+1} = \boldsymbol{N}\boldsymbol{\xi}_k + dt(\mathbb{I}\otimes\boldsymbol{R}_{op}^{\top})vec(\dot{\boldsymbol{R}}_k). \tag{3.24}$$

The discrete dynamics in $\boldsymbol{\xi}$ is obtained by putting in (3.20) and pre-multiply with $\boldsymbol{N}^{*}$, the left pseudo-inverse of $\boldsymbol{N}$

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + dt\boldsymbol{N}^*(\mathbb{I} \otimes \boldsymbol{R}_{op}^\top)(\boldsymbol{C}_\xi^c + \boldsymbol{C}_\xi^\xi \boldsymbol{\xi}_k + \boldsymbol{C}_\xi^\omega \boldsymbol{\omega}_k). \tag{3.25}$$

The angular velocity dynamics in (3.16) is also vectorized. The second term in (3.16) is $\delta \boldsymbol{R}_k^\top \boldsymbol{\tau}_{op} = (\mathbb{I} \otimes \boldsymbol{\tau}_{op}^\top)\boldsymbol{N}\boldsymbol{\xi}_k$ and $\delta\boldsymbol{\tau}_k$ is defined in (3.17). The last two terms can be further derived,

$$\begin{aligned} &-\delta\hat{\boldsymbol{\omega}}_k {}^B\boldsymbol{I}\boldsymbol{\omega}_{op} - \hat{\boldsymbol{\omega}}_{op} {}^B\boldsymbol{I}\delta\boldsymbol{\omega}_k \\ &\quad = (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op} {}^B\boldsymbol{I})\delta\boldsymbol{\omega} \\ &\quad = (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op} {}^B\boldsymbol{I})(\boldsymbol{\omega}_k - \boldsymbol{\omega}_{op} - \hat{\boldsymbol{\omega}}_{op}\boldsymbol{\xi}_k). \end{aligned} \tag{3.26}$$

Assembling these expressions into (3.16) and rearranging the corresponding terms gives the vectorization of ${}^B\boldsymbol{I}\dot{\boldsymbol{\omega}}_k$

$${}^B\boldsymbol{I}\dot{\boldsymbol{\omega}}_k = \boldsymbol{C}_{\dot\omega} + \boldsymbol{C}_{\dot\omega}^{\delta p}\boldsymbol{p}_k + \boldsymbol{C}_{\dot\omega}^{\xi}\boldsymbol{\xi}_k + \boldsymbol{C}_{\dot\omega}^{\omega}\boldsymbol{\omega}_k + \boldsymbol{C}_{\dot\omega}^{\delta u}\delta\boldsymbol{u}_k, \tag{3.27}$$

where

$$\begin{aligned} \boldsymbol{C}_{\dot\omega}^c &= -\hat{\boldsymbol{\omega}}_{op} {}^B\boldsymbol{I}\boldsymbol{\omega}_{op} + \boldsymbol{R}_{op}^T\boldsymbol{\tau}_{op} - (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op} {}^B\boldsymbol{I})\boldsymbol{\omega}_{op} \\ &\quad - \boldsymbol{R}_{op}^\top(\sum \hat{\boldsymbol{u}}_{op})\boldsymbol{p}_{op} \\ \boldsymbol{C}_{\dot\omega}^{\delta p} &= \boldsymbol{R}_{op}^\top(\sum_i \hat{\boldsymbol{u}}_{op}^i) \\ \boldsymbol{C}_{\dot\omega}^{\xi} &= (\mathbb{I} \otimes \boldsymbol{\tau}_{op}^\top)\boldsymbol{N} - (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op} {}^B\boldsymbol{I})\hat{\boldsymbol{\omega}}_{op} \\ \boldsymbol{C}_{\dot\omega}^{\omega} &= \widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op} {}^B\boldsymbol{I} \\ \boldsymbol{C}_{\dot\omega}^{\delta u} &= \boldsymbol{R}_{op}^\top[\hat{\boldsymbol{r}}_{op}^1, \hat{\boldsymbol{r}}_{op}^2, \hat{\boldsymbol{r}}_{op}^3, \hat{\boldsymbol{r}}_{op}^4]. \end{aligned} \tag{3.28}$$

The discrete dynamics of $\boldsymbol{\omega}$ is propagated using forward Euler scheme $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + dt\dot{\boldsymbol{\omega}}_k$.

3.2.4 Discrete-time Affine Dynamics

The single rigid body model introduced in Section 3.2.1 has nonlinear dynamics in $\boldsymbol{R}$ and $\boldsymbol{\omega}$. Hence, a variation-based linearization scheme is proposed in Section 3.2.2 to linearize the nonlinear dynamics. Section 3.2.3 presents a vectorization method to reformulate the matrix variables into the vector form. Based on the aforementioned procedures, the new set of state and control vectors are defined as,

$$\begin{aligned} \boldsymbol{x}_k &:= [\boldsymbol{p}_k^\top \;\; \dot{\boldsymbol{p}}_k^\top \;\; \boldsymbol{\xi}_k^\top \;\; {}^B\boldsymbol{\omega}_k^\top]^\top \in \mathbb{R}^{12} \\ \delta\boldsymbol{u}_k &:= [\delta\boldsymbol{u}_{1,k}^\top \;\; \delta\boldsymbol{u}_{2,k}^\top \;\; \delta\boldsymbol{u}_{3,k}^\top \;\; \delta\boldsymbol{u}_{4,k}^\top]^\top \in \mathbb{R}^{12}, \end{aligned} \tag{3.29}$$

where the new control input $\delta\boldsymbol{u}_i \in \mathbb{R}^3$ is the variation of GRF from the operating point $\boldsymbol{u}_{i,op}$ for the i^{th} leg.

By assembling the corresponding terms from (3.25), (3.27) into matrix form, the discrete-time affine dynamics can be expressed in the state-space form:

$$\boldsymbol{x}_{t+k+1|t} = \boldsymbol{A}|_{op} \cdot \boldsymbol{x}_{t+k|t} + \boldsymbol{B}|_{op} \cdot \delta\boldsymbol{u}_{t+k|t} + \boldsymbol{d}|_{op}, \tag{3.30}$$

where $\boldsymbol{A}|_{op} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B}|_{op} \in \mathbb{R}^{n\times m}$, and $\boldsymbol{d}|_{op} \in \mathbb{R}^n$ are matrices constructed by the measurements at the operating point. Therefore, nonlinear dynamics have been linearized about the operating point to result in a locally-valid linear time-varying (LTV) system. This system can be stabilized to track reference trajectories.

The discrete-time affine dynamics are imposed as equality constraints as in (3.5b).

3.2.5 Cost Function

The cost function in the nonlinear MPC formulation (3.5) includes both terminal and stage costs. In this work, the cost is set as a quadratic function that penalizes deviation from the reference trajectories. The stage cost is

$$\ell(\boldsymbol{x}_k, \boldsymbol{u}_k) = ||\boldsymbol{x}_k - \boldsymbol{x}_{d,k}||^2_{\boldsymbol{Q}_x} + ||\boldsymbol{u}_k - \boldsymbol{u}_{d,k}||^2_{\boldsymbol{R}_u}, \tag{3.31}$$

where $||\boldsymbol{x}||^2_{\boldsymbol{Q}}$ is a shorthand notation of the matrix norm $\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x}$ where $\boldsymbol{Q}$ is a positive definite matrix; $\boldsymbol{x}_{d,k}$ and $\boldsymbol{u}_{d,k}$ are the desired state and control at the k^{th} prediction step; $\boldsymbol{Q}_x$ and $\boldsymbol{R}_u$ are the block diagonal positive definite gain matrices for state and control, respectively. The first term in (3.31) consisting of the error functions of the state vector can be decomposed as:

$$||\boldsymbol{x}_k - \boldsymbol{x}_{k,d}||^2_{\boldsymbol{Q}} = ||\boldsymbol{e}_{p_k}||^2_{\boldsymbol{Q}_p} + ||\boldsymbol{e}_{\dot{p}_k}||^2_{\boldsymbol{Q}_{\dot{p}}} + ||\boldsymbol{e}_{R_k}||^2_{\boldsymbol{Q}_R} + ||\boldsymbol{e}_{\omega_k}||^2_{\boldsymbol{Q}_\omega}, \tag{3.32}$$

where $\boldsymbol{Q}_p, \boldsymbol{Q}_{\dot{p}}, \boldsymbol{Q}_R, \boldsymbol{Q}_\omega$ are diagonal positive definite weighting matrices; $\boldsymbol{e}_{p_k}, \boldsymbol{e}_{\dot{p}_k}$ are the error terms for deviations from the corresponding reference trajectories $\boldsymbol{p}^d_k, \dot{\boldsymbol{p}}^d_k$ constructed from the user input. The error function for the rotation matrix and angular velocity are given by [17] as

$$\boldsymbol{e}_{R_k} = \log(\boldsymbol{R}^\top_{d,k} \cdot \boldsymbol{R}_k)^\vee \tag{3.33}$$

$$\boldsymbol{e}_{\omega_k} = \boldsymbol{\omega}_k - \boldsymbol{R}^\top_k \boldsymbol{R}_{d,k}\boldsymbol{\omega}_{d,k}, \tag{3.34}$$

where $\boldsymbol{R}_{d,k}$ and $\boldsymbol{\omega}_{d,k}$ are the desired rotation matrix and angular velocity trajectories. The terminal cost function is similarly defined.

In the stage cost expression (3.32), all the error terms are in the linear form of the state and control vectors except the error of orientation $\boldsymbol{e}_{R_k}$, which involves matrix logarithm map as shown in (3.33). A linear approximation of the nonlinear error term on the rotation matrix is used. Taking the hat map on (3.33) and applying the matrix exponential map give

$$\exp(\hat{\boldsymbol{e}}_{R_k}) = \boldsymbol{R}^\top_{d,k}\boldsymbol{R}_k \approx \boldsymbol{R}^\top_{d,k}\boldsymbol{R}_{op}\exp(\hat{\boldsymbol{\xi}}_k), \tag{3.35}$$

where the same approximation is made here as in (3.9). Taking the matrix logarithm of

(3.35) and then applying the vee map give the approximate error term on $\boldsymbol{R}_k$,

$$\boldsymbol{e}_{R_k} = \log(\boldsymbol{R}_{d,k}^\top \cdot \boldsymbol{R}_{op})^\vee + \boldsymbol{\xi}_k. \tag{3.36}$$

The error function defined in (3.36) is in linear form of the state variable $\boldsymbol{\xi}_k$ since both $\boldsymbol{R}_{d,k}^\top$ and $\boldsymbol{R}_{op}$ are known matrices. The error function can be interpreted as the sum of (a) the geodesic between $\boldsymbol{R}_{op}$ and $\boldsymbol{R}_{d,k}$ and (b) the vector $\boldsymbol{\xi}_k$ which lies in the tangent space at $\boldsymbol{R}_{op}$. The orientation error function Ψ on $\boldsymbol{R}$ can therefore be defined as,

$$\Psi(\boldsymbol{\xi}_k) = \boldsymbol{e}_{R_k}^\top \boldsymbol{Q}_R \boldsymbol{e}_{R_k} = ||\boldsymbol{e}_{R_k}||_{\boldsymbol{Q}_R}^2, \tag{3.37}$$

which is in quadratic form of $\boldsymbol{\xi}_k$. Given that the weighting matrix $\boldsymbol{Q}_R$ is positive definite, the orientation error function Ψ is positive definite.

The terminal cost ℓ_T is similarly defined as (3.32) with terminal gains. The cost of control is constructed as

$$||\boldsymbol{u}_k - \boldsymbol{u}_{d,k}||_{\boldsymbol{R}_u}^2 = ||\boldsymbol{u}_{op} + \delta\boldsymbol{u}_k - \boldsymbol{u}_{d,k}||_{\boldsymbol{R}_u}^2, \tag{3.38}$$

where $\delta\boldsymbol{u}_k$ is the k^{th} predicted variation from the operating point $\boldsymbol{u}_{op}$.

3.2.6 Force Constraints

The force constraints are imposed to ensure that the solved GRF are physically feasible. When the foot is in contact with the ground, the normal force should be non-negative and the tangent forces should lie within the friction cone, which is prescribed as

$$\{\boldsymbol{u}_i | \boldsymbol{u}_i^n \geq 0, ||\boldsymbol{u}_i^t||_2 \leq \mu |u_i^n|\}, \tag{3.39}$$

where μ is the coefficient of friction; superscript $(\cdot)^t$ and $(\cdot)^n$ indicate tangential and normal force components, respectively. $||\cdot||_2$ is the 2-norm and $|\cdot|$ takes the absolute value of a scalar.

Since the friction cone constraint is a second-order cone constraint, it is not admissible to the QP formulation with linear constraints. Instead, the conservative friction pyramid [133] is used as an approximation of the friction cone. In addition, the normal force is bounded to ensure that the commanded torque does not exceed the actuator limits. The feasible control set $\mathbb{U}$ in (3.5) is defined as:

$$\begin{aligned}\mathbb{U}_i := \{\delta \boldsymbol{u}_i \mid |u^{x/y}_{i,op} + \delta u^{x/y}_{i,k}| &\leq \mu |u^z_{i,op} + \delta u^z_{i,k}|, \\ u^{z,min}_{i,k} &\leq u^z_{i,op} + \delta u^z_{i,k} \leq u^{z,max}_{i,k}, \\ u^{z,min}_{i,k} &\geq 0\},\end{aligned} \tag{3.40}$$

where the z axis is aligned with the normal vector of the ground and x, y are two axes orthogonal to each other that lie in the tangent plane at the contact point; $u^{z,min}_{i,k}$ and $u^{z,max}_{i,k}$ are the minimum and maximum normal forces at the k^{th} predicted step for leg i. If leg i was in swing phase, then the value for both lower and upper bounds are set to zero so that the swing leg controller takes over and guides the foot towards the next foothold position. It can be observed that (3.40) denotes an intersection of a finite set of closed halfspaces in $\mathbb{R}^3$. Hence, $\mathbb{U}_i$ is a polyhedron. Similarly, the feasible force set $\mathbb{U}$ is also polyhedral.

3.2.7 Quadratic Program Formulation

Given the convex quadratic cost function from Section 3.2.5, the affine dyanmics from Section 3.2.4 and linear force constraints from Section 3.2.6, the general nonlinear MPC problem

(3.5) can be be reformulated as a Quadratic Program (QP),

$$\begin{aligned}
\text{min.} \quad & \gamma^N \ell_T(\boldsymbol{x}_{t+N|t}) + \sum_{k=0}^{N-1} \gamma^k \ell(\boldsymbol{x}_{t+k|t}, \delta \boldsymbol{u}_{t+k|t}) && \text{(3.41a)} \\
\text{s.t.} \quad & \boldsymbol{x}_{t+k+1|t} = \boldsymbol{A}\boldsymbol{x}_{t+k|t} + \boldsymbol{B}\delta\boldsymbol{u}_{t+k|t} + \boldsymbol{d} && \text{(3.41b)} \\
& \delta\boldsymbol{u}_{t+k|t} \in \mathbb{U}_k, k = 0, \cdots, N-1 && \text{(3.41c)} \\
& \boldsymbol{x}_{t|t} = \boldsymbol{x}(t) = \boldsymbol{x}_{op}, && \text{(3.41d)}
\end{aligned}$$

where the cost function is defined in (3.31); the decay rate factor $\gamma \in (0, 1]$ discounts cost further from the current moment. The definition of the affine dynamics can be found in (3.30) and the force constraint is expressed in (3.40). Note that we lifted the explicit constraints on the state vectors but instead relied on the cost function for the state regulation. The QP in (3.41) can be rewritten in a more compact form. Following the formulation in [139], the new optimization variable $\boldsymbol{z}$ is constructed as

$$\boldsymbol{z} = [\delta\boldsymbol{u}_0^\top, \boldsymbol{x}_1^\top, \cdots, \delta\boldsymbol{u}_{N-1}^\top, \boldsymbol{x}_N^\top]^\top \in \mathbb{R}^{24N}, \tag{3.42}$$

such that (3.41) can be transcribed into the standard QP form [15],

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\boldsymbol{z}^\top \boldsymbol{P}\boldsymbol{z} + \boldsymbol{c}^\top \boldsymbol{z} \\
\text{subject to} \quad & \boldsymbol{A}_{ineq} \cdot \boldsymbol{z} \leq \boldsymbol{b}_{ineq} \\
& \boldsymbol{A}_{eq} \cdot \boldsymbol{z} = \boldsymbol{b}_{eq},
\end{aligned} \tag{3.43}$$

where $\boldsymbol{P} \in \mathbb{R}^{N(n+m)}$ is a symmetric positive definite matrix assembled from the gain matrices $\boldsymbol{Q}_x, \boldsymbol{R}_u$; the inequality constraint $\boldsymbol{A}_{ineq} \cdot \boldsymbol{z} \leq \boldsymbol{b}_{ineq}$ imposes the force constraints; the equality constraint $\boldsymbol{A}_{eq} \cdot \boldsymbol{z} = \boldsymbol{b}_{eq}$ respects the linear dynamics.

3.3 Numerical Implementation

This section presents the simulation results of RF-MPC stabilizing various periodic gaits and an aperiodic 3D acrobatic maneuver. Furthermore, RF-MPC is compared with the MPC that uses Euler angles as orientation representation (EA-MPC) in the acrobatic maneuver. The resulting QPs from the MPC formulation in all of the simulations are solved using MATLAB *quadprog*. Gain values and gait pattern parameters for the following simulations can be found in Table B.1.

3.3.1 Walking Trot

The data of a walking trot simulation is shown in Fig. 3.3. The robot starts from a stationary pose and accelerates in the x-direction until it reaches the final velocity of 0.5 m/s. As can be seen in Fig. 3.3 (a), the velocity in the x-direction reaches 0.5 m/s and the velocity deviation for all directions is within ± 0.1 m/s. Fig. 3.3 (b) shows that the orientation deviation in all directions is bounded within ± 0.02 rad. The vertical GRFs of all four legs are shown in Fig. 3.3 (c). Further details about the generation of the reference trajectory for trotting can be found in Section 3.3.5.

The simulation is set up such that at each sampling time, the control input is applied to the original nonlinear model (3.8) simulated using MATLAB *ode*45 to integrate the dynamics. The gait pattern in the walking trot simulation is executed using a time-based schedule.

3.3.2 Bounding

To demonstrate the capability of RF-MPC to stabilize dynamic motions with large body attitude oscillation, the bounding simulation is presented. The bounding motion involves an aerial phase when all four feet lose contact with the ground. To stabilize the bounding motion, the reference trajectory is designed based on the impulse-scaling principle [107] to

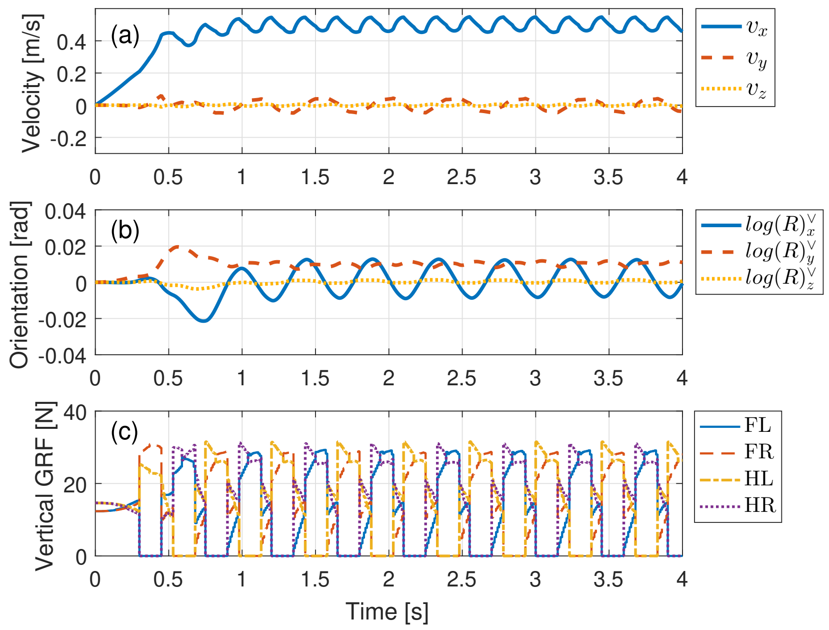


Figure 3.3: Simulation results of walking trot where the robot starts from static pose and accelerates in the x-direction (a) CoM velocity; the robot accelerates at 0.5 m/s^2 and reaches the desired velocity of 0.5 m/s in the x-direction. (b) Orientation in terms of the log map of the rotation matrix (c) Vertical ground reaction forces of four legs.

preserve the periodicity of the gait. Details about the generation of reference trajectory can be found in Section 3.3.5. Here, we kindly note that an elaborate reference trajectory is optional for RF-MPC to stabilize bounding. While a trivial reference such as that used for trotting also works, the reference trajectory presented in Section Section 3.3.5 enables bounding motions with a longer aerial phase. Similar to the walking trot, the robot is commanded to start from the static pose and accelerates at 1.0 m/s^2 to reach the final velocity of 2.5 m/s.

Fig. 3.4 (a) presents the phase portrait of angle and angular velocity along the y-axis. It shows that the motion converges to a periodic orbit. The velocity tracking performance shown in Fig. 3.4 (b) demonstrates that the MPC controller is capable of stabilizing bounding velocity up to 2.5 m/s. The vertical GRF profiles of legs FL and HL are shown in Fig. 3.4 (c).

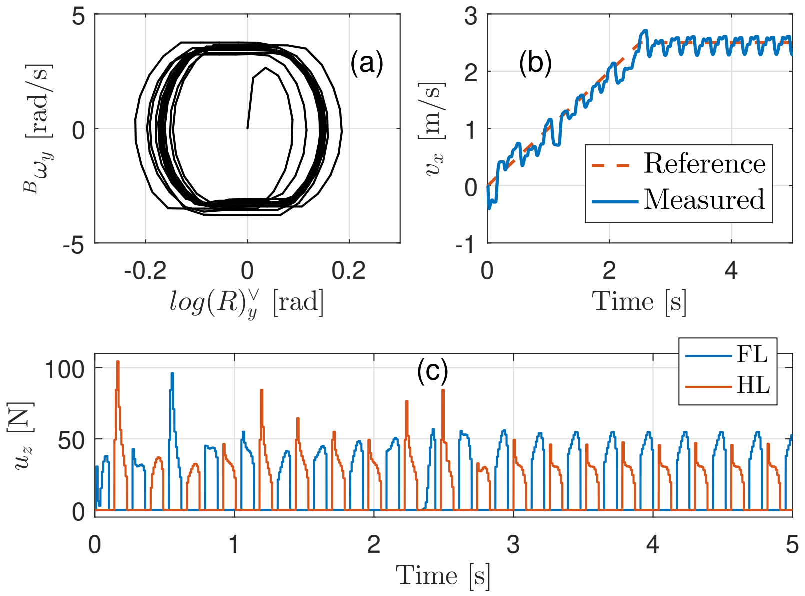


Figure 3.4: Simulation results of bounding. (a) Phase portrait of body angle and angular velocity in the y-axis. (b) Velocity tracking performance in the x-direction. (c) Vertical GRFs of FL and HL legs.

3.3.3 Aperiodic Complex Dynamic Maneuver

A complex acrobatic dynamic maneuver is presented in this section to demonstrate that RF-MPC is capable of controlling aperiodic dynamic motions that involve orientations that correspond to singularities in Euler angle representation. RF-MPC is benchmarked with an MPC controller with Euler angles for its orientation representation. In addition, the initial condition is perturbed to investigate the robustness of RF-MPC.

Fig. 3.5 (a) shows the reference trajectory of the acrobatic motion, which is a backflip with a twist. The reference CoM trajectory is colored black and the reference poses are shown in blue. The robot initially stands on a slope with the slope angle of 45°, with the front of the robot facing upwards and body parallel to the slope. The stance phase of this acrobatic jump consists of 0.1 s of all four feet in contact, followed by 0.1 s of only the hind feet pair in contact with the slope. After the stance phase, all four feet are airborne and the robot enters the aerial phase for 0.3 s before landing. The landing direction of the robot is facing away from the slope. The feed-forward GRF and the dynamically-feasible reference trajectory are generated by solving an off-line TO problem, where the slope is set to be 45°. Further details about the generation of the reference trajectory are provided in Section 3.3.5.

As can be observed from Fig. 3.5 (a), when the robot approaches the singularity, EA-MPC becomes unstable and exerted a large vertical force that pushes the robot away from the reference trajectory. As shown in Fig. 3.5 (c) (d), the body orientation and CoM position eventually diverge from the reference trajectory after the robot encounters singularity, which is visualized in Fig. 3.5 (e). In comparison, Fig. 3.5 (b) shows that RF-MPC can successfully stabilize the backward tumble motion that involves singularity. Here, we would like to point out in 3.5 (e) that the robot actually passes through singularity when the hind legs are in contact as indicated by the duration when $\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}}) < 10^{-1}$, in the light shaded area. Fig. 3.5 (c) (d) display that the orientation and CoM position deviation are kept small during the motion. The CoM position and orientation deviations are defined as

$$\begin{aligned} |\boldsymbol{e}_p| &= |\boldsymbol{p}(t) - \boldsymbol{p}_d(t)| \\ |\boldsymbol{e}_R| &= |log(\boldsymbol{R}_d^\top(t)\boldsymbol{R}(t))^\vee|, \end{aligned} \tag{3.44}$$

where $\boldsymbol{p}_d$ and $\boldsymbol{R}_d$ are the reference CoM position and body orientation, respectively.

To demonstrate the robustness of RF-MPC, the slope angle for the simulated cases is changed from 45° to 53.6°. Since the body of the robot is parallel to the slope at the beginning of the jump, the initial orientation of the robot is also perturbed. As can be observed from Fig. 3.5 (c), the backflip can be executed and stabilized by RF-MPC. In contrast, an open-loop simulation shows that in the absence of feedback control, the orientation of the robot quickly deviates from the reference trajectory due to the initial condition perturbation.

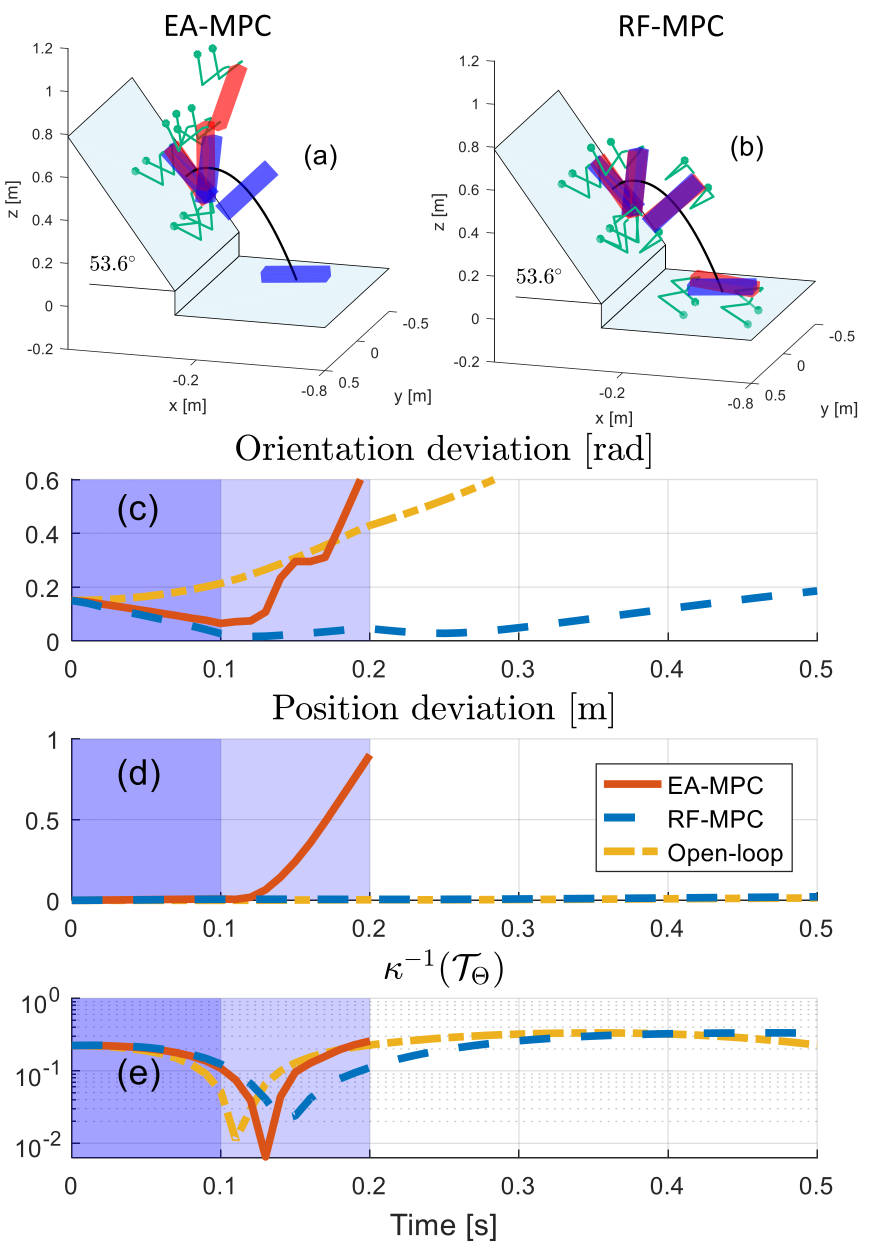


Figure 3.5: Simulation results of a complex aperiodic 3D maneuver where the robot performs a twisting jump off an inclined surface. The reference poses are shown in blue and the poses controlled by MPC are shown in red; the reference CoM trajectory is shown in black. The deep-shaded area is when four legs are in contact with the surface; the light-shaded area is when only hind legs are in contact and the non-shaded area is when the robot is in the aerial phase. (a) The EA-MPC becomes unstable when the robot is close to the singular pose. (b) RF-MPC can track the reference motion that involves singular poses. (c) The orientation deviation $|\boldsymbol{e}_R|$ of open-loop control, RF-MPC, and the EA-MPC. (d) The CoM position deviation. (e) The function to quantify the distance to singularity $\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}})$.

3.3.4 Comparison of Linearization Schemes

One of the crucial decisions we made in the proposed RF-MPC is to linearize the dynamics about the operating point. The choice is made because RF-MPC represents orientation using the rotation matrix, which presumes $SO(3)$ structure. Such a structure loses its validity when the predicted states are far away from the point where the linearization is performed upon. Nevertheless, a reasonable alternative is linearizing around the reference trajectory, which is a widely used technique in robotics. To investigate which linearization scheme provides more robust behavior, this section presents a simulation case study that compares MPC linearized around the reference trajectory (scheme 1) with MPC linearized around the operating point (scheme 2).

Scheme 1 linearizes dynamics around the reference trajectory, which includes the desired state $\{\boldsymbol{x}^d_{t+k|t}\}$ and control $\{\boldsymbol{u}^d_{t+k|t}\}$ within the prediction horizon, where $k = 0, \cdots, N-1$ and N is the prediction horizon. Hence, $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ are matrices for a Linear Time-Varying (LTV) system, parametrized by the reference trajectory within the prediction horizon. Scheme 2 linearizes dynamics around the operating point, which involves current state $\boldsymbol{x}_{op}$ and control $\boldsymbol{u}_{op}$, as defined in Section 3.2. Constant matrices $\boldsymbol{A}|_{op}$ and $\boldsymbol{B}|_{op}$ are used to propagate the state through the prediction horizon using a Linear Time-Invariant (LTI) system.

The robustness of these two linearization schemes is qualitatively compared by examining how much external disturbance they can handle. The simulation is set up such that robot is bounding at a constant speed of 1.0 m/s in the $+x$ direction. The disturbance with a maximum force of 27 N is applied to the robot in the $+y$ direction, causing it to deviate from the reference trajectory. The reference trajectory, controller gain, gait timing, and disturbance are the same for both schemes, with only the linearization scheme being different.

Fig. 3.6 (a) shows a sequential snapshot of the simulated scenario for scheme 2. The GRFs are shown in red and the disturbance force (visible at t = 1.0 s and t = 1.5 s) is in cyan. Fig. 3.6 (b) shows the disturbance force profile, which is applied at the FR shoulder

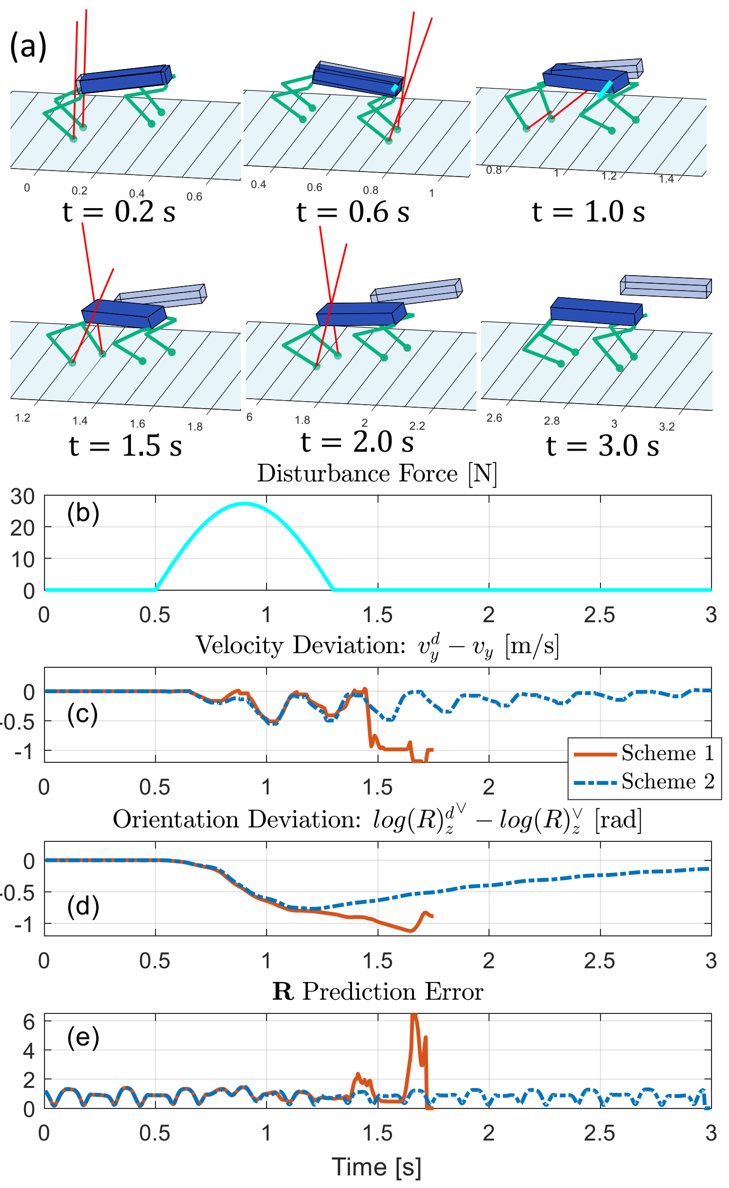


Figure 3.6: Comparison between two linearization schemes. (a) A sequential snapshot of the simulation scenario for scheme 2. The GRFs are shown in red and the disturbance force is shown in cyan. The translucent box represents the reference pose of the robot. (b) Disturbance force profile. (c) Velocity deviation and (d) Orientation deviation in the y-direction from the reference trajectory, respectively. (e) Prediction error of the Rotation matrix.

of the robot.

RF-MPC using scheme 1 fails at 1.5 s since the velocity and orientation start to diverge from the reference, as shown in Fig. 3.6 (c) and (d), respectively. In comparison, the RF-MPC using scheme 2 recovers from the disturbance and successfully tracks the reference trajectory. To investigate the reason for the discrepancy, we defined a quantity that measures the prediction quality of the rotation matrices,

$$\Psi(t) = \sum_{k=1}^{N} ||\widetilde{\boldsymbol{R}}_k - \overline{\boldsymbol{R}}_k||_F + ||log(\boldsymbol{R}_{op,t+k}^\top \widetilde{\boldsymbol{R}}_k)^\vee||, \tag{3.45}$$

where $\boldsymbol{R}_{op,t+k} \in SO(3)$ is the rotation matrix at time $t+k \cdot dt$; $\overline{\boldsymbol{R}}_k$ is the k^{th} predicted rotation matrix at time t, whose projection to the $SO(3)$ manifold is denoted as $\widetilde{\boldsymbol{R}}_k \in SO(3)$; $||\cdot||_F$ is the Frobenius norm of a matrix. The prediction error of rotation matrices is plotted in Fig. 3.6 (e), where the error value of scheme 1 became high when the system started to deviate from the reference trajectory. This numerical study serves as an empirical validation of the robustness of scheme 2 in comparison to scheme 1 in disturbance rejection.

3.3.5 Reference Trajectory Generation

Trotting

The reference control trajectory $\boldsymbol{u}_{k,d}$ for trotting is defined based on the heuristic that the total weight of the robot is supported evenly by all the contact legs:

$$u_{i,k,d}^{x/y} = 0, \;\; u_{i,k,d}^{z} = \frac{b_{i,k}}{\sum_{i=1}^{4} b_{i,k}} Mg, \tag{3.46}$$

where $b_{i,k} \in \{0, 1\}$ is a binary variable that indicates the contact condition of leg i at instance $t+k$, where $b_{i,k} = 1$ indicates contact phase and 0 otherwise. The value of the binary variable $b_{i,k}$ is defined according to the time schedule of a Finite State Machine (FSM), which is introduced in Section 3.4.1. The reference state $\boldsymbol{x}_{k,d}$ is constructed by simply assuming the robot accelerate from static pose with constant acceleration until reaching the maximal velocity. Both walking trot and running trot use the same reference trajectory.

Aperiodic Complex Dynamic Maneuver

The reference trajectory is generated by an off-line TO algorithm based on the single-shooting method. The twist jump motion is decomposed into three phases with fixed timing. Phase one is when four feet are in contact, which lasts for 0.1 s; phase two is when front legs lift-off and hind legs are in contact, which lasts for 0.1 s; phase three is when the robot is airborne, which lasts for 0.3 s. The optimization variables are the magnitude of the GRFs, which is assumed to be constant throughout phases one and two. The cost function is the weighted sum of the control effort and the deviation from the desired landing pose. The constraints imposed in the optimizations are

- Fixed initial state and bounded final state
- Fixed contact sequence and timing
- Kinematic reachability of each leg
- Collision avoidance with the environment
- Unilateral GRF stay within friction cone.

The TO is formulated as a nonlinear program with 24 variables, representing the force magnitude of 4 GRFs (each has 3 components) in two stance phases. The optimization problem is solved by the MATLAB NLP solver *fmincon*.

Bounding

The periodic trajectory for bounding gait is generated by considering the robot as a single rigid body in 2D, which has 3 DoFs (x, z, θ). The contact sequence and timing is pre-specified as front-stance, aerial phase I, hind-stance and aerial phase II. The shapes of vertical GRF and pitch torque profiles are parametrized by Bézier polynomials. Periodicity in the z and θ directions is achieved by finding the scaling factors and initial condition based on the principle of impulse-scaling [107] analytically.

Controlled Tumble

To generate the reference trajectory of the controlled tumble in Section 3.5.4, we used the open-source *OptimTraj* library [69] to set up the TO problem using the direct collocation method. The optimization is done on a 2-D single rigid body model of the robot. The convex quadratic cost function penalizes large GRF and rewards smooth force profiles. In addition to the constraints mentioned in Section 3.3.5, the following constraints are also imposed

- The constraints that enforce dynamic feasibility.
- Path constraints on the state and control.

The above problem setup has 27 time steps, which results in an optimization problem with 272 variables and 366 constraints solved by MATLAB NLP solver *fmincon*.

3.4 Control Framework

The MPC framework in Section 3.2 is combined with other components such as state estimation and swing leg controllers to give rise to various motions implemented on the robot hardware platform. This section presents the implementation details that are required to realize the MPC control design on the hardware.

Fig. 3.7 shows the schematics of the overall control system. The Finite State Machine (FSM) sends the desired state and control trajectories $\boldsymbol{X}_d, \boldsymbol{U}_d$ to the MPC, which formulates a Quadratic Program (QP). The QP is solved by the custom QP solver qpSWIFT [106] to obtain the optimal solution $\delta \boldsymbol{u}$, which is added to the control at the operating point $\boldsymbol{u}_{op}^-$ to get the GRF $\boldsymbol{u}_{op}$. A swing leg controller calculates the swing force $\boldsymbol{u}_{sw}$ to track the swing

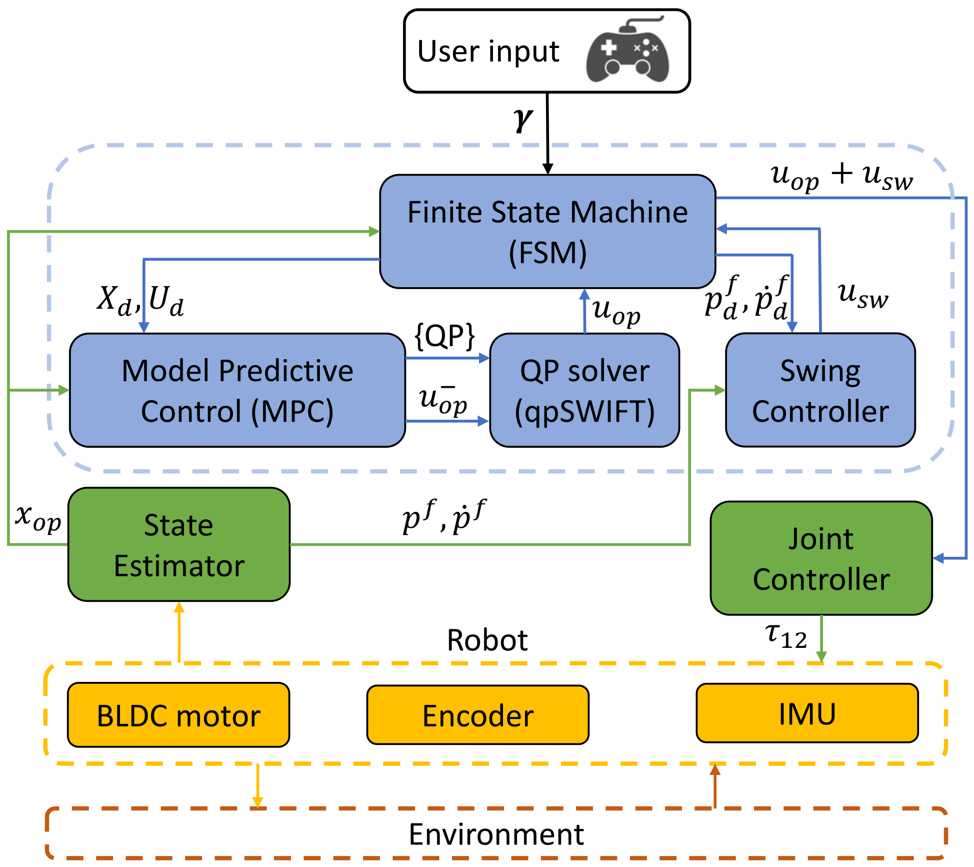


Figure 3.7: Overview of the control system. The user sends commands to the on-board computer (blue), where the finite state machine schedules the gait and sends desired trajectories to the MPC block to formulate the QP. The custom QP solver qpSWIFT solves for the $\boldsymbol{u}_{op}$ and send it to the FSM. The FSM combines the stance and swing forces and sends to the joint controller (green), which maps leg forces to joint torque and sends to the BLDC motors. The state estimator (green) receives sensor signals for the MPC formulation of the next cycle. The previous solution of the QP $\boldsymbol{u}_{op}^-$ is sent to the MPC as the control at the operating point.

foot trajectories. The commanded torque is modified by a lower-level joint controller, which compensates for friction and motor dynamics. The Brush-Less Direct Current (BLDC) motors actuate the robot to interact with the environment. The QP solver qpSWIFT [106] is designed to efficiently solve MPC problems with the decision variable size of around 200.

3.4.1 Finite State Machine

Various gaits are generated by a finite state machine (FSM). Fig. 3.8 shows the schematics of the FSM where the timing schedules are sent from the gait planner to each leg. A leg

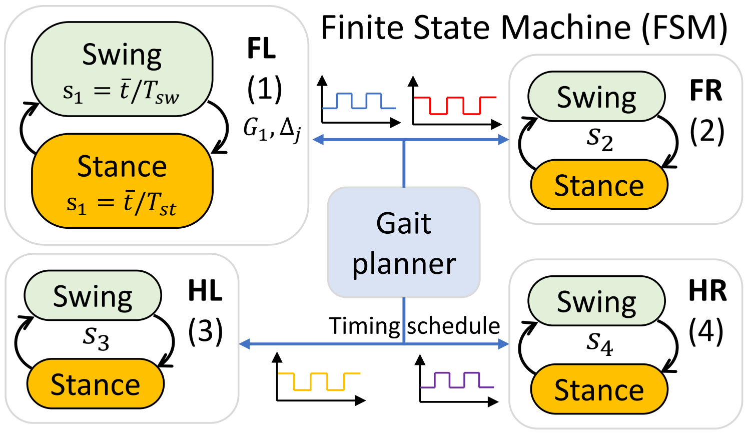


Figure 3.8: Schematics of the Finite State Machine (FSM). The gait planner sends to all legs the timing schedules; the normalized variable s_i is the percentile completion of the current state. $\Delta_j, j \in \{st, sw\}$ are the reset maps and G_i are the guard sets.

independent phase variable s_i quantifies the percentile completion of either stance or swing state. The phase variables are defined as $s_i := \{\bar{t}/T_j \text{ s.t. } j \in \{st, sw\}\}$, where $\bar{t}$ represents the current dwell time, T_{st}, T_{sw} are stance and swing times, respectively. The period of the gait is the sum of the dwell times $T = T_{st} + T_{sw}$. The guard sets G_i and reset maps Δ_j define the transition between states. The guard sets are given as $G_i := \{\bar{t} \text{ s.t. } \bar{t} = T_j\}$. The reset map is defined as $\Delta_j(\bar{t}) = 0$ such that it resets the phase variable and current dwell time. This framework allows the implementation of any gait sequence by changing the timing schedules. The contact detection algorithm can be incorporated to adjust the gait timings and extend the time-based FSM to event-based FSM. Using the FSM scheme, trotting, bounding, and aperiodic motions can be realized. It is worth noting that the prediction horizon can cover multiple phases. Hence, in motions with aerial phases such as bounding and acrobatic jump, the RF-MPC can take into consideration of the upcoming phase change and plan the current control accordingly.

3.4.2 Computation Setup

The MPC framework is implemented using Simulink Real-Time (SLRT). The encoder readings and lower-level kinematics calculations are carried out at a base rate of 4 kHz, while the IMU signals are received and state estimation is performed at 1 kHz. The user input from the joystick is updated at 23 Hz, and the QP is solved at between 160 Hz to 250 Hz depending on the size of the problem. The proposed QP (3.43) is solved by a custom QP solver qpSWIFT [106] for all the experiments. Written in ANSI C, the solver is library-free while and it interfaces with SLRT through a gateway *s-function*. An RF-MPC with a prediction horizon of 6 entails solving a QP with 144 variables, 72 inequality, and 72 equality constraints.

3.5 Experiment Results

The proposed RF-MPC controller is a general motion control framework that can be used to achieve multiple motion objectives. This section presents the experimental results of some common tasks for quadrupedal robots, including pose control, balancing on a moving platform, and periodic locomotive gaits such as walking trot, running trot, and bounding. In addition, a controlled tumble experiment is presented to show that the RF-MPC framework is capable of controlling dynamic motions previously hard to achieve because of the presence of singularity. The gain values and the gait timing for all experiments can be found in Table B.2.

3.5.1 Pose and Balancing Control

To exhibit the tracking performance of the MPC controller, the pose control experiment is conducted. The experimenter sends the desired CoM vertical height and orientation commands in y and z directions to the robot from the joystick. The MPC controller continuously solves for the desired GRFs at the four feet, which are in contact with the ground throughout the experiment. The position and orientation reference tracking data shown in Fig. 3.9 suggests that RF-MPC can closely track the reference command. To demonstrate the capability of RF-MPC to balance its body under large disturbances, the balancing experiment is

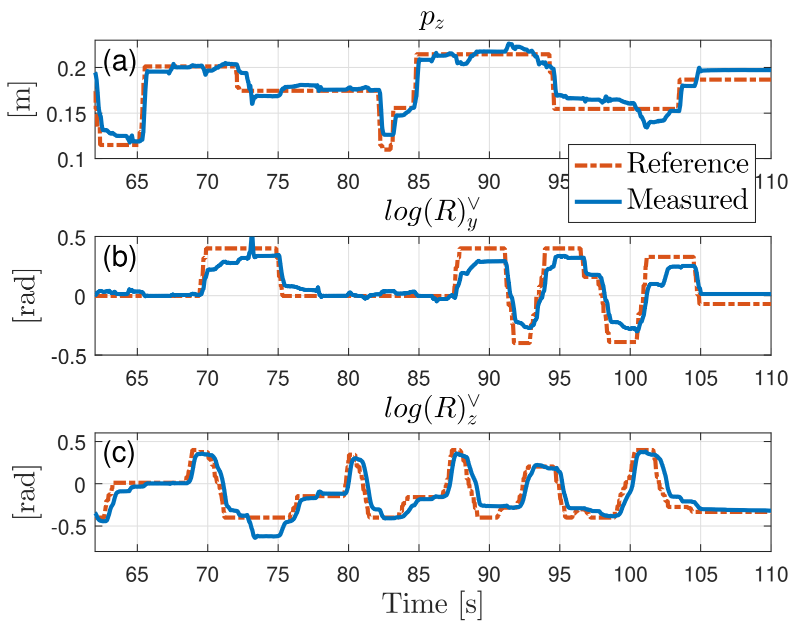


Figure 3.9: Pose control experiment data. (a) the position tracking performance for p_z, (b) and (c) present the orientation tracking performance in the y and z directions.

presented. The experimental setup is shown in Fig. 3.10 (a). The robot stands on a pivoted platform, attempting to maintain the balance at the nominal standing pose when the platform is perturbed by the operator. The robot body coordinate $\{B\}$ and the coordinate of the platform $\{P\}$ are both plotted in Fig. 3.10 (a). The origin of $\{P\}$ is set at the center of the four feet. Fig. 3.10 (b) shows the orientation deviation in the x and y directions for $\{P\}$ in blue and $\{B\}$ in red; Fig. 3.10 (c) shows the angular velocity in the x and y directions. As shown in Fig. 3.10, the balancing controller significantly reduces the movement of the robot's body frame $\{B\}$ compared to that of the platform-fixed frame $\{P\}$.

3.5.2 Walking Trot

To demonstrate that RF-MPC can stabilize basic locomotion gaits, the walking trot experiment is presented. The robot can move in any direction parallel to the ground while maintaining the body orientation. Fig. 3.11 (a) and (b) exhibit the velocity tracking performance of the controller, and Fig. 3.11 (c) shows that the orientation deviation is kept small

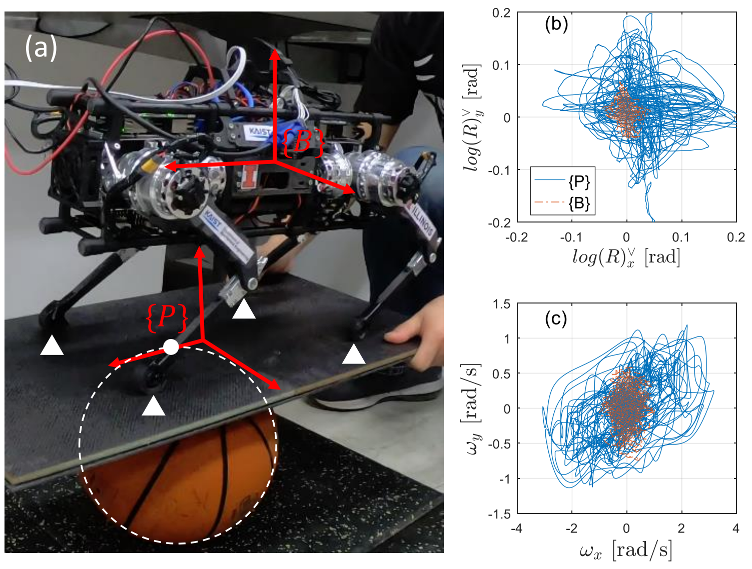


Figure 3.10: The balancing control experiment (a) Experimental setup. The robot stands on a platform pivoted on a sphere, the pivot point is shown as a solid circle. The four triangles indicate the foot contact points. (b) The orientation deviation of the platform coordinate $\{P\}$ (blue) and the body coordinate $\{B\}$ (red). (c) The body angular velocity of the platform coordinate (blue) and the body coordinate (red).

(within ± 0.06 rad) during the walking trot experiment; Fig. 3.11 (d) presents the vertical GRF during the walking trot. The velocities are measured from the state estimation.

3.5.3 Running Trot and Bounding

To investigate the performance of RF-MPC for dynamic gaits, experiments of running trot and bounding gaits with full aerial phases are conducted. Fig. 3.12 presents the running trot experiment data, where Fig. 3.12 (a) shows that the vertical CoM velocity experiences 40 ms free fall during the aerial phase. Fig. 3.12 (b) shows that the robot can produce abruptly changing GRF as the contact condition changes. During the trot running experiment, the vertical GRF can reach as high as 60 N while the knee torque goes up to 6.3 Nm, as can be observed in Fig. 3.12(b) and (c), respectively.

The bounding gait leverages the full dynamics of the robot and involves extensive body pitch oscillation. Sequential snapshots of the bounding experiment can be found in Fig. 3.13 (a). The robot starts from a static pose and the MPC stabilizes the robot to follow

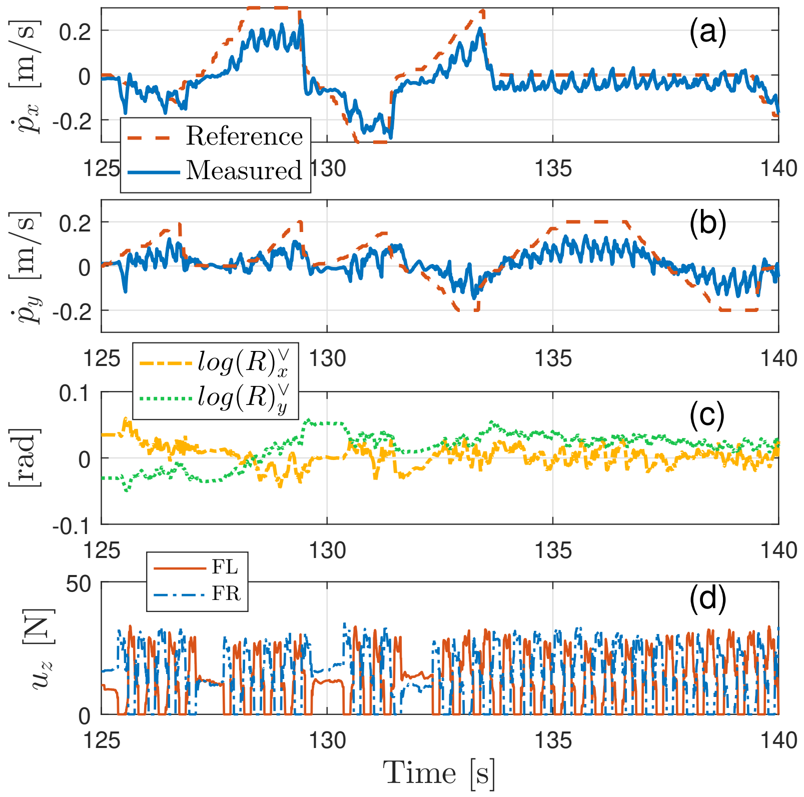


Figure 3.11: Walking trot experiment data. (a) Velocity tracking in the x-direction. (b) Velocity tracking in the y-direction (c) Orientation deviations along the x and y-axes, where the reference is 0. (d) Commanded vertical GRF u_z for front legs.

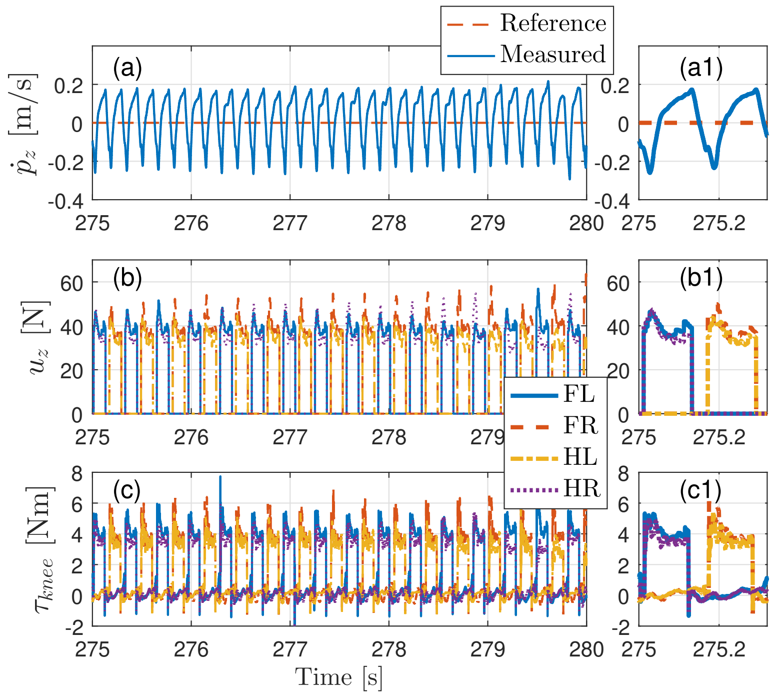
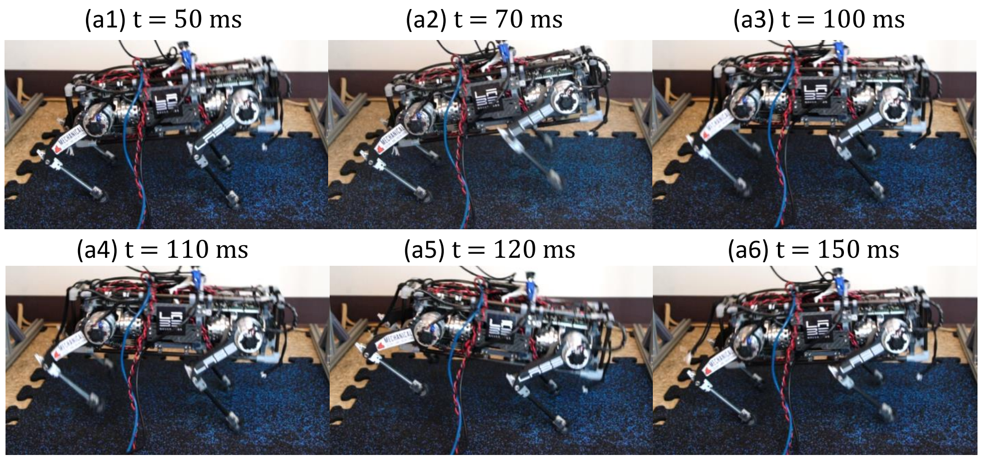


Figure 3.12: Running trot experiment data. Zoomed-in views placed at the right of the figure show the details of the signals. (a) Reference and measured CoM vertical velocity in the z-direction (b) Vertical GRF (c) Knee torque.

the desired GRF and state trajectories. More details about reference trajectory generation can be found in Section 3.3.5. The reference and measured trajectories of orientation and angular velocity in the y-direction are shown in Fig. 3.13 (b), (c). Since the robot started from a static pose, there is an initial offset. The vertical GRF profile is shown in Fig. 3.13 (d). The transition from the swing to the stance phase occurs when a touchdown event is declared by the contact detection algorithm described in Section 2.4.3.

3.5.4 Controlled Backflip

To demonstrate the capability of RF-MPC to control dynamic maneuvers that involve singularity poses, a controlled tumble experiment is presented. As shown in Fig. 3.14, the robot flips backward around the y-axis, passing through the pose where the robot is upright, before it lands with the upside-down orientation. Note that even though the reference trajectory is



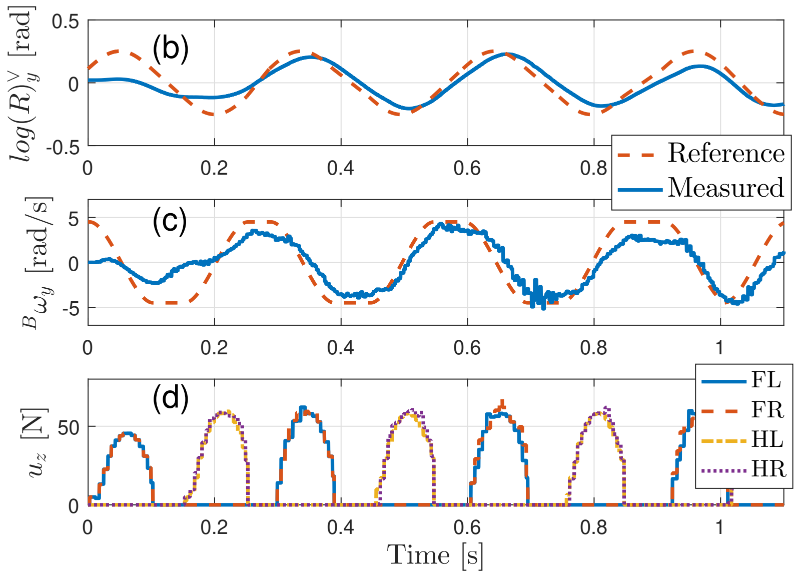


Figure 3.13: Bounding experiment results (a) Sequential snapshots of the robot in a bounding experiment. (b) Orientation tracking in the y-direction $log(R)_y^\vee$ (c) Angular velocity tracking in the y-direction $^B\omega_y$ (d) Vertical GRF.

generated based on a 2-D model of the robot as presented in Section 3.3.5, RF-MPC controls the 3D robot in the controlled tumble experiment without resorting to the decomposition of sagittal plane motion and out-of-plane motion.

The image sequence in Fig. 3.15 (a) is plotted along with the body orientation, which is reconstructed from the experiment data as shown in Fig. 3.15 (b). The rotation matrix is represented by the body coordinate frame axes (x-blue, y-red, z-green); the three color-coded rings correspond to the Euler angles with the roll-pitch-yaw sequence convention (roll-blue, pitch-red, yaw-green). Note that the robot is at the singular pose at around 300 ms as shown in Fig. 3.15 (a3). In the corresponding body orientation plot, the axes of the rings for Euler angles almost coincide. Therefore, the robot indeed passes through the singularity pose when the RF-MPC is actively controlling the hind legs to track the reference trajectory. To the best of the authors' knowledge, this is the first instance of hardware experimental implementation of MPC to control acrobatic motion which involves singularity.

Fig. 3.16 presents the data from the controlled tumble experiment, where the robot goes through three phases. The deep-shaded area corresponds to the phase when all four legs are in contact; the light-shaded area indicates the phase when only the hind legs are in contact; the non-shaded area corresponds to the landing phase. The RF-MPC controller is activated during the first two phases, and an impedance control is utilized in the third phase. Experiment data gathered from 10 backflip trials are shown in Fig. 3.16, where the solid lines are the mean values of all the tests, and the shaded tube is the value within one standard deviation.

As can be observed from Fig. 3.16 (d), the robot passes through the neighborhood of singularity as the number introduced in Section 3.1.1 drops below the threshold 10^{-1}. Fig. 3.16 (c) shows that the pitch angle θ is not monotonic throughout the controlled tumble while $log(R)^\vee_y$ decreases monotonically. The dash-dot curves in Fig. 3.16 (a) and (c) are from the experiment trial where the initial state of the robot is perturbed. Specifically, the height of the stage on which the front legs are positioned is increased from 80 mm to 130

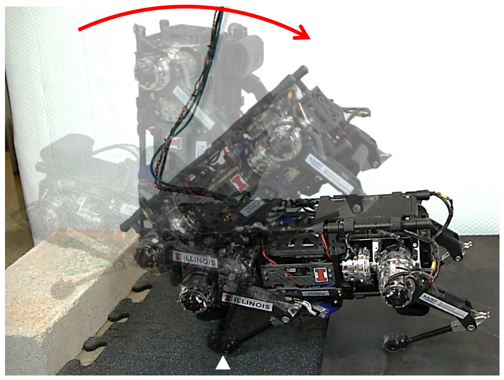


Figure 3.14: Quadruped robot *Panther* performing a controlled tumble that involves passing through the upright pose, which corresponds to the singularity in the Euler angle representation. Throughout the controlled tumble, the feet pair indicated by the white triangle is kept in contact with the ground. The red arrow indicates the direction of the backflip and the shadowed images are snapshots during the backflip.

mm. It can be observed that while in this case, the trajectory of the robot deviates more than one standard deviation from the average, RF-MPC can still stabilize the motion and land safely.

3.6 Summary

In this chapter, we presented a representation-free model predictive control framework that directly represents orientation using the rotation matrix instead of using other orientation representations. Despite the local validity of linearized dynamics on the rotation matrix, this approach introduces the possibility to stabilize 3D complex acrobatic maneuvers that involve singularities in the Euler angles formulation. By directly working on the rotation matrix, this method avoids issues arising from the usage of other representations such as unwinding phenomenon (quaternion) or singularity (Euler angles). The application of a variation-based linearization scheme and a vectorization routine linearized the nonlinear dynamics and transformed the matrix variables into vector variables. The deliberate construction of the orientation error function enabled us to formulate the MPC into the standard QP form.

We reported both simulation and experiment results of the RF-MPC controller applied

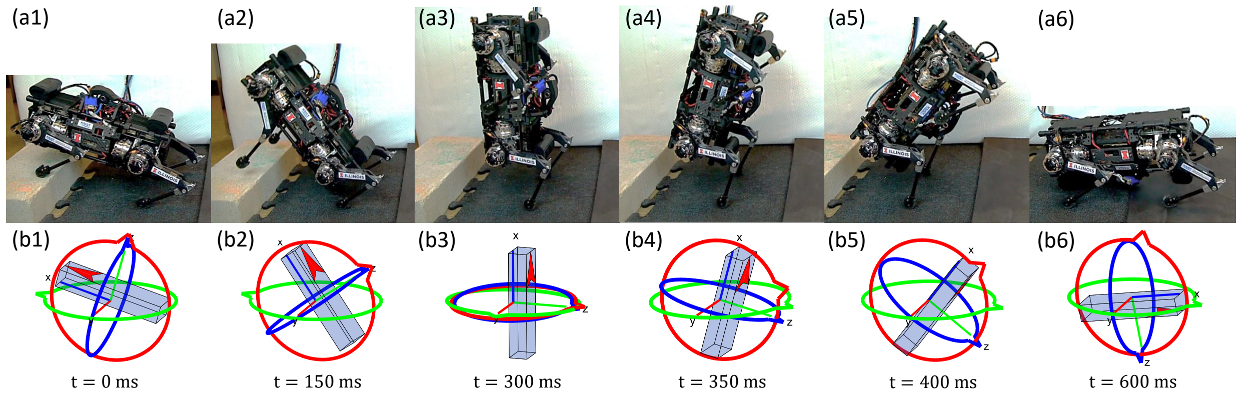


Figure 3.15: Quadruped robot *Panther* performing a controlled tumble that passes through singularity pose. (a) An image sequence of the robot executing the controlled tumble, with its front legs launching from an 80 mm high platform. (b) The body orientation reconstructed from the experimental data. The rotation matrix is represented by the body coordinate frame axes (x-blue, y-red, z-green); the Euler angles are visualized by the three colored rings with arrows (ϕ-blue, θ-red, ψ-green). The robot passes through the upright pose (a3) while the hind legs are in contact with the ground. The Gimbal lock effect is shown in (b3) where axes of Euler angles are aligned.

on the quadruped *Panther* robot. In the simulation case study presented in Section 3.3.4 we found out that in the RF-MPC framework, linearizing around the operating point provides a more robust control strategy compared with linearizing around the reference trajectory. Experiments including pose/balance control, walking/running trot, and bounding were conducted on the robot. In addition, the controlled tumble experiment demonstrated that the RF-MPC controller can stabilize dynamic motions that involve singularity. By utilizing a custom QP solver qpSWIFT, the MPC can reach a control frequency as high as 250 Hz.

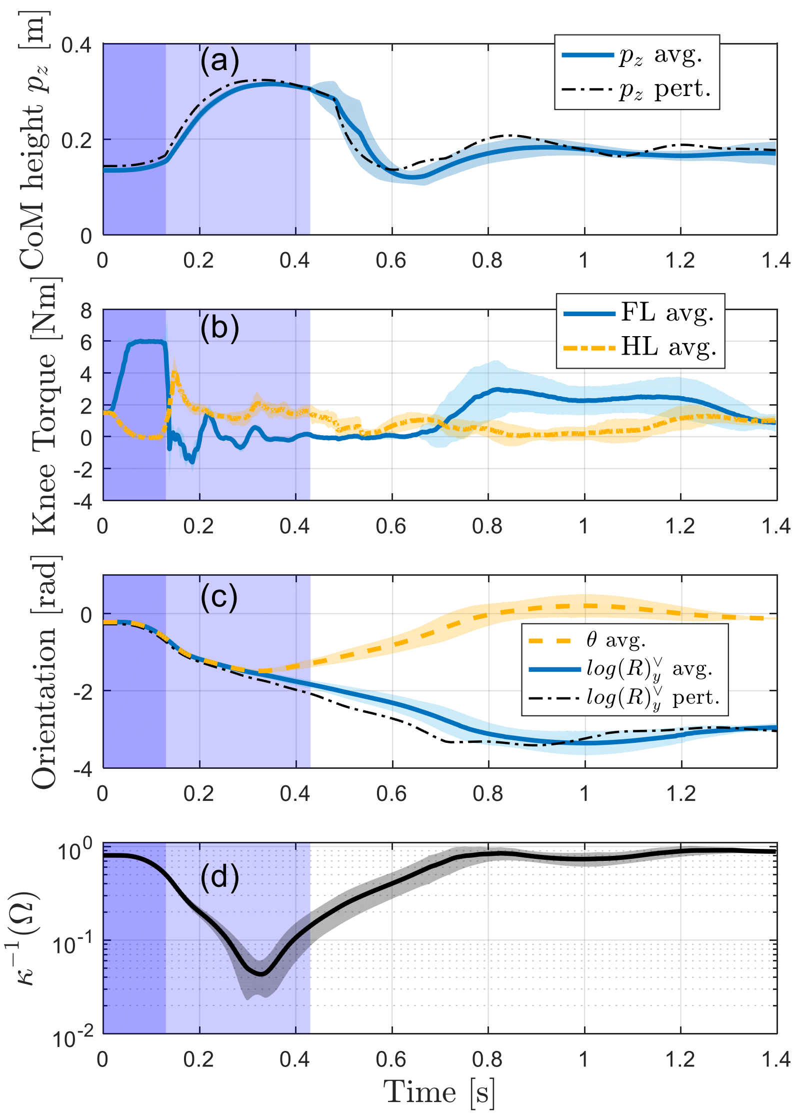


Figure 3.16: Experimental data from 10 controlled tumble trials. The solid lines are the average (avg.) of all the tests, and the shaded tube is the range within one standard deviation. The black dash-dot curves in (a) and (c) are from the case where the initial condition of the backflip is perturbed (pert.). The deep-shaded area is when four legs are in contact; the light-shaded area is when hind legs are in contact, and the non-shaded area is when all legs are in the impedance control phase. (a) The CoM height. (b) The knee torque of legs FL and HL. (c) Comparison between the pitch angle θ and rotation matrix $log(R)_y^\vee$. (d) The function $\kappa^{-1}(\mathcal{T}_\Theta)$ indicates that the robot indeed encountered the singular pose in the controlled tumble experiment.

Chapter 4

Kinodynamic Motion Planning via Mixed-Integer Convex Program

This chapter presents a mixed-integer convex program (MICP) formulation for kinodynamic motion planning problems for dynamic legged robots to traverse challenging terrains. This motion planning algorithm simultaneously reasons about centroidal motion, contact location, actuator torque limit, and gait sequence in a single MICP. Specifically, the non-convex actuator constraint is relaxed to piece-wise convex constraints over a discretization of the configuration space. In addition, the bilinear terms are approximated by McCormick envelope convex relaxation. The MICP can be efficiently solved to the global optimum by off-the-shelf numerical solvers and provide highly dynamic jumping motions without initial guesses. Simulation and experimental results demonstrate that the proposed method can find novel and dexterous maneuvers that are directly deployable on a single-legged and a two-legged robot to overcome challenging terrains[1]. This chapter is based on our work in [31] and [30].

4.1 Introduction

The ability to perform dynamic motions such as leaping over gaps and jumping on high platforms is a unique advantage of legged systems. Planning for dynamic motions such as jumping is a challenging problem since it involves both continuous and discrete variables. This problem requires decision-making in a semi-continuous search space, which involves continuous variables describing robot state, contact positions, and contact wrenches; It also

[1]Video clips featuring the simulation and experiment results are available in movie1 and movie2.

involves discrete variables such as the foothold position and gait sequence.

Many methods have been developed to solve this problem. For example, the trajectory optimization (TO) approach locally improves upon an initial motion plan by solving a general nonlinear optimization problem using a gradient-based nonlinear solver. There has been tremendous progress in using TO to solve locomotion problems. MIT Cheetah 2 robot can jump over obstacles by solving nonlinear constrained optimization online [107]. Optimized jumping trajectories are generated offline [101] and implemented on MIT Cheetah 3 [11]. Linear complementary problems (LCP) are formulated in [111] to generate trajectories without *a priori* contact scheduling. Dynamic movements without scheduled contact are also generated in [87] using a hierarchical framework. The combined planning problem is solved in [95] by incorporating all constraints into the objective function, and solve unconstrained nonlinear programming (NLP). Legged locomotions with gait sequences are generated on non-flat terrain in [143] in a single TO formulation using a phase-based parameterization method. These methods either rely on explicit contact schedules or require solving a large NLP. The size and non-convexity of these problems imply that the nonlinear solver is only effective in searching for a local minimum around the initial guess. Hence, proper initialization of the optimization is crucial in finding a feasible solution. Besides, the infeasible status returned by a local NLP solver is not informative since one is not sure whether the planning problem itself is infeasible or it is not initialized properly.

4.1.1 Motivation

Mixed-integer convex optimization does not rely on the initial seed and warrants global solution [15]. With the recent advancement in numerical solvers, a medium-sized mixed-integer convex programming (MICP) can be solved efficiently by off-the-shelf solvers such as gurobi [102], Mosek [97] and CPLEX [63]. Due to its feature that warrants a global solution with either global optimality or infeasibility certificate, MICP has found many applications in robotics. For example, it has been used in global inverse kinematics [25],

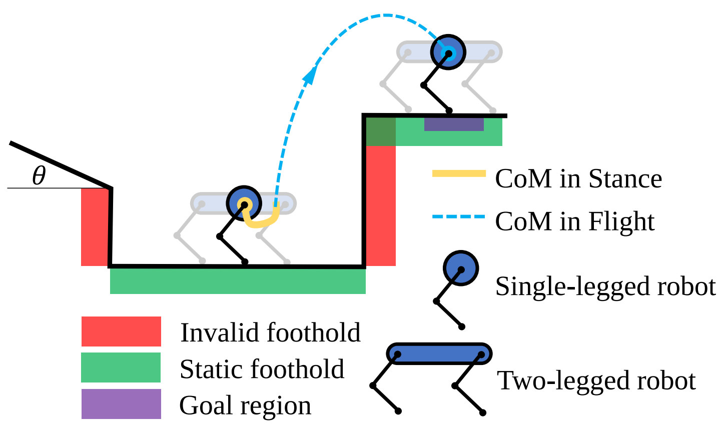


Figure 4.1: Illustration of the kinodynamic motion planning problem for jumping legged robots. The robot systems of concern include a planar single-legged robot (R1) and a planar two-legged robot (R2). The terrain height variation is higher than the robot height, so the robot has to execute jumping motions to reach the goal region.

grasping [82], footstep planning [28], quadruped locomotion planning [1], and aggressive legged locomotion [134], [31]. The distinct feature of mixed-integer programs (MIP) is that the binary variables can turn on/off constraints, which enables reasoning about discrete choices.

However, the combinatorial nature of MIP causes the *curse of dimensionality*, which means a MIP quickly becomes intractable as the number of variables increases. The issue of high computational requirements is mitigated by exploiting the structure of each individual problem. Specifically, simplified models and condensed constraint formulations are employed to reduce the number of variables used in the MICP. Because of the problem-specific nature of the MICP formulation, two case studies are presented where MICP comes up with dynamic motions for a planar single-legged robot and a planar two-legged robot.

4.1.2 Problem Statement

The examples included in this chapter only concern planar robots, which are tasked to start from an initial position to reach the goal region. The height variation of the terrain is larger than the height of the robots, hence, the robot has to execute jumping motions to overcome obstacles and reach the goal region. As shown in Fig. 4.1, the terrain is modeled as a piecewise affine function, where the goal region is colored purple. The jumping motions of the robots compose of alternating stance phases and aerial phases. Unlike the single-legged robot whose stance phase is unambiguous, the stance phase of the two-legged robot can be further sub-categorized into the front, back, and double stance. For conciseness, the single-legged robot will be referred to as R1, and the two-legged robot R2 for the rest of the chapter.

To solve the kinodynamic motion planning problem, a TO problem such as the following can be formulated

$$\underset{\boldsymbol{x}_{opt}}{\text{minimize}}\ f_0(\boldsymbol{x}_{opt}) \tag{4.1a}$$

$$\text{subject to}\ \dot{\boldsymbol{X}} = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{u}) \tag{4.1b}$$

$$\boldsymbol{p}_{fp} \in Terrain \tag{4.1c}$$

$$\boldsymbol{q}(t) \in \boldsymbol{\Omega} \tag{4.1d}$$

$$\boldsymbol{q}(t_0) \in \boldsymbol{Q}_0 \tag{4.1e}$$

$$\boldsymbol{q}(t_f) \in \boldsymbol{Q}_g \tag{4.1f}$$

$$\boldsymbol{u}(t) \in \boldsymbol{U}(\boldsymbol{q}(t)) \tag{4.1g}$$

where $\boldsymbol{x}_{opt}$ is the decision variable vector; $f_0(\cdot)$ is a convex objective function. $\boldsymbol{X} = [\boldsymbol{q}^\top, \dot{\boldsymbol{q}}^\top]^\top$ is the state vector, where $\boldsymbol{q}$ is the configuration of the robot and $\dot{\boldsymbol{q}}$ is its time derivative. $\boldsymbol{f}(\cdot)$ is the continuous time dynamics and $\boldsymbol{u}$ is the control vector. $\boldsymbol{p}_{fp}$ is the concatenated foothold position vector, which should be on the $Terrain$ surface during stance phases. Constraint (4.1d) limits $\boldsymbol{q}$ to be within the configuration space (C-space) $\boldsymbol{\Omega}$ of the robot; t_0 and t_f refer to the initial and final time; $\boldsymbol{Q}_0$ and $\boldsymbol{Q}_g$ represents the initial configuration set and goal configuration set, respectively. $\boldsymbol{U}$ indicates the set of admissible control as a function of the configuration of the robot.

4.2 Common Formulations for R1 and R2

This section collects all the common formulations for the single-legged robot R1 and the two-legged robot R2, although the specific implementation differs due to platform differences. For example, the simplified dynamic of R1 and R2 takes the same double integrator form, but the dimension is different since R1 is modeled as a point-mass and R2 has an additional pitch dimension.

4.2.1 Reduced Model

To simplify the robot dynamics, the leg mass is neglected and the equation of motion takes the form of a double integrator

$$\ddot{\boldsymbol{q}} = \boldsymbol{\mathcal{D}}^{-1}\boldsymbol{u} + \boldsymbol{a}_g \tag{4.2}$$

where $\boldsymbol{q}$ is the configuration of the robot; $\boldsymbol{\mathcal{D}}$ is the inertia matrix; $\boldsymbol{u}$ is the control input vector; $\boldsymbol{a}_g$ is the gravitational acceleration vector. Here, R1 is modeled as a point-mass and R2 as a single rigid body, where the centroidal dynamics [103] are used to capture the dynamics of the robot systems. Therefore, the input to R1 is the GRF $\boldsymbol{u} \in \mathbb{R}^2$, and the input to R2 is the spatial wrench about CoM $\boldsymbol{u} \in \mathbb{R}^3$. The expressions for each term in Eq. (4.2) are summarized in the following table for robots R1 and R2.

Table 4.1: Definition of each term in the simplified dynamics Eq. (4.2) for both robots, as shown in Fig. 4.1

Terms	robot R1	robot R2
$\boldsymbol{q}$	$[x, z]^\top$	$[x, z, \theta]^\top$
$\boldsymbol{u}$	$[F_x, F_z]^\top$	$[F_x, F_z, \tau_y]^\top$
$\boldsymbol{\mathcal{D}}$	$\text{diag}(m, m)$	$\text{diag}(m, m, I_\theta)$
$\boldsymbol{a}_g$	$[0, -g]^\top$	$[0, -g, 0]^\top$
$dim(q)$	2	3
N_{leg}	1	2

In table 4.1, $[x, z]^\top$ is the CoM position; θ is the pitch angle; F_x, F_z are the sum of GRF in the x and z directions, respectively; τ_y is the effective moment applied at CoM. m is the total mass of the robot, and I_θ is the moment of inertia; $\text{diag}(\cdot)$ creates a diagonal matrix from the input vector; g is the constant of gravitational acceleration; $dim(q)$ is the dimension of the configuration space; N_{leg} is the total leg number of the robot.

Let the control trajectory $\boldsymbol{u}(t)$ be parameterized by the Bézier polynomial of degree N_B with coefficient $\boldsymbol{\alpha}_u$. Then, the state trajectories $\dot{\boldsymbol{q}}(t), \boldsymbol{q}(t)$ are also Bézier polynomials with coefficients $\boldsymbol{\alpha}_{\dot{q}}$ and $\boldsymbol{\alpha}_q$, respectively. Given initial conditions $\dot{\boldsymbol{q}}_0, \boldsymbol{q}_0$, $\boldsymbol{\alpha}_{\dot{q}}$ and $\boldsymbol{\alpha}_q$ can be calculated by linear operations

$$\boldsymbol{\alpha}_{\dot{q}} = \boldsymbol{\mathcal{L}}(\boldsymbol{\alpha}_u, \dot{\boldsymbol{q}}_0), \quad \boldsymbol{\alpha}_q = \boldsymbol{\mathcal{L}}(\boldsymbol{\alpha}_{\dot{q}}, \boldsymbol{q}_0), \tag{4.3}$$

where the linear operation $\boldsymbol{\mathcal{L}}(\cdot)$ is defined in Appendix 6.3.

The continuous time trajectories $\boldsymbol{u}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{q}(t)$ are sampled at a discrete time sequence $\{t_i | i = 1, 2, \cdots, N_t\}$, where N_t is the number of nodes during the stance phase.

4.2.2 Configuration Space

The configuration space (C-space) of the robot is the set of configurations that the robot can reach during the stance phase without violating kinematic constraints. The set of constraints that define the C-space is

$$\boldsymbol{\Omega} := \{\boldsymbol{q} \in \mathbb{R}^{dim(q)} \mid \boldsymbol{q}_{lb} \leq \boldsymbol{q} \leq \boldsymbol{q}_{ub} \tag{4.4a}$$

$$\textbf{for } i = 1 \textbf{ to } N_{leg} \tag{4.4b}$$

$$\hat{\boldsymbol{n}}^\top \cdot (\boldsymbol{p}_i^h - \boldsymbol{p}_i^f) \geq 0 \tag{4.4c}$$

$$\hat{\boldsymbol{n}}^\top \cdot (\boldsymbol{p}_i^k - \boldsymbol{p}_i^f) \geq 0 \tag{4.4d}$$

$$\textbf{if } \boldsymbol{p}_i^f \textbf{ in contact} \tag{4.4e}$$

$$|\boldsymbol{p}_i^h - \boldsymbol{p}_i^f|_2 \in [r_{lb}, r_{ub}]\}, \tag{4.4f}$$

where constants $dim(q)$ and N_{leg} are given in Table 4.1 for robot R1 and R2. $\hat{\boldsymbol{n}}$ is the unit normal vector of the terrain; $\boldsymbol{p}^h, \boldsymbol{p}^k, \boldsymbol{p}^f$ are the hip joint, knee joint and foot positions. (4.4a) is an element-wise box constraint on the robot configuration. Inequalities (4.4c) and (4.4d) prohibit the hip and knee from penetrating the ground; (4.4f) limits the leg extension to be within the range $[r_{lb}, r_{ub}]$.

4.2.3 Aerial Phase Kinematics

The robot is assumed to execute a series of jumping motions, which involves both stance and aerial phases. The kinematic relationship between the take-off state of the previous stance and the touch-down state of the subsequent stance is

$$\begin{bmatrix} \bar{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \end{bmatrix}_{j+1}^{TD} = \begin{bmatrix} \bar{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \end{bmatrix}_{j}^{TO} + \begin{bmatrix} \dot{\boldsymbol{q}}_j^{TO} \\ \boldsymbol{a}_g \end{bmatrix} T_{j,air} + \begin{bmatrix} \frac{1}{2}\boldsymbol{a}_g \\ \boldsymbol{0} \end{bmatrix} T_{j,air}^2 \tag{4.5}$$

where the superscripts $(\cdot)^{TO}$ and $(\cdot)^{TD}$ indicate variables at take-off and touch-down, respectively; $T_{j,air}$ is the aerial time of the i^{th} jump; where $j = 1, \cdots N_j$ and N_j is the number of jumps. The state $\bar{\boldsymbol{q}}$ is the sum of the foothold configuration and the local configuration $\bar{\boldsymbol{q}} = \boldsymbol{q}_{fp} + \boldsymbol{q}$, where the foothold configuration $\boldsymbol{q}_{fp}$ is used to select the terrain segment for the next stance. Meanwhile, configuration space constraint and wrench constraint can be imposed on the local configuration $\boldsymbol{q}$.

4.2.4 Bilinear Terms

The aerial phase kinematic equation (4.5) is non-convex because $\dot{\boldsymbol{q}}^{TO} \cdot T_{air}$ is a bilinear term. The bilinear term can be approximated using the McCormick Envelope [90], which approximates the saddle-shaped bilinear product surface $x \cdot y$ as a collection of convex polytopes. The McCormick envelope formulation transcribes the originally non-convex blinear constraints into mixed-integer convex ones. This approach was used in [134] for planning aggressive motions of legged robots, and it is adopted here to preserve the mixed-integer convex formulation. Since the range of each quantity of the bilinear product can be obtained empirically from simulations and experiments, a relatively small number of binary variables are needed for the McCormick Envelope. Similarly, the quadratic term T_{flt}^2 in (4.5) is approximated by a piecewise affine function. The binary variables used in the approximation of the bilinear and quadratic terms are collected in the binary matrix $\boldsymbol{B}^{mc}$.

4.2.5 Foothold Position Choices

In this chapter, the terrain is modeled as a piecewise affine function. A binary matrix variable $\boldsymbol{B}^{fp} \in \{0, 1\}^{N_s \times N_j}$ can be constructed to assign foothold positions, where N_s is the total number of terrain segments and N_j is the total number of jumps. $\boldsymbol{B}^{fp}_{s,j} = 1$ implies that at the jump j, the foothold position lies on the segment s,

$$\boldsymbol{B}^{fp}_{s,j} = 1 \implies \boldsymbol{p}^{j}_{fp} \in seg_s, \tag{4.6}$$

where the *implies* operator ($\implies$) in (4.9) is implemented using the big-M formulation [119]. The big-M formulation represents piece-wise affine constraints using binary variables. Other methods such as the convex-hull approach [132] can be used here, but the big-M formulation yields a problem with fewer decision variables. $\boldsymbol{p}_{fp}^{j}$ is the foothold position for the j^{th} jump. The terrain segment seg_s is a line segment defined as

$$seg_s = \{\boldsymbol{p} = [x, z]^\top \in \mathbb{R}^2 | \boldsymbol{A}_s \cdot \boldsymbol{p} \leq \boldsymbol{b}_s, x \in [x_s^{lb}, x_s^{ub}]\}, \tag{4.7}$$

where $\boldsymbol{A}_s$ and $\boldsymbol{b}_s$ describe the affine function with domain $[x_s^{lb}, x_s^{ub}]$.

To ensure that for each jump, the foothold position only lands on exactly one of the terrain segments, the following constraint has to be imposed.

$$\sum_{s=1}^{N_s} \boldsymbol{B}_{s,j}^{fp} = 1, \forall j = 1, \cdots, N_j \tag{4.8}$$

Note that Eq. (4.8) is a mixed-integer linear constraint.

4.3 Planar Single-Legged Robot R1

This section presents how the kinodynamic motion planning problem of R1 can be formulated as a MICP. Specifically, the non-convex torque limit constraint relaxed as mixed-integer ellipsoidal constraints, making the resulting mathematically program a Mixed-Integer Quadratically Constrained Program (MIQCP). The torque limit constraint in this chapter is assumed to be not velocity-dependent to make the problem more tractable. The experiment result of a Parkour motion is presented to validate the proposed formulation.

4.3.1 C-Space Discretization

The C-space of R1 as shown in Fig.4.2(a) is discretized into $N_d \in \mathbb{Z}^+$ number of triangle cells $\boldsymbol{c}_k \subset \boldsymbol{\Omega}, k \in \{1, \cdots, N_d\}$. The union of polytopes $\boldsymbol{c}_i$ constitutes an inner approximation

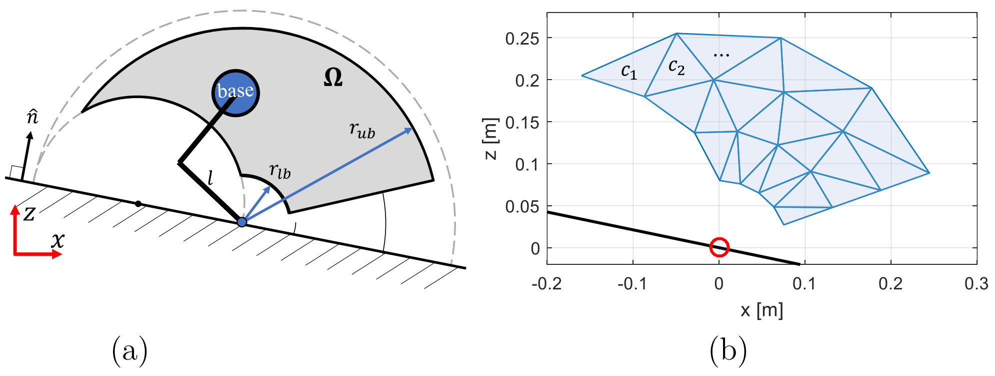


Figure 4.2: Illustrations of the C-space of the planar single-legged robot R1 (a) The original C-space (b) An example of the C-space discretization with $N_d = 24$ cells

of $\boldsymbol{\Omega}$. Fig. 4.2(b) shows an example of workspace discretization, where $N_d = 24$. A binary matrix $\boldsymbol{B}^{cs} \in \{0,1\}^{N_d \times N_t}$ is constructed such that $\boldsymbol{B}^{cs}_{k,i} = 1$ indicates that $\boldsymbol{q}(t_i)$ is within $\boldsymbol{c}_k$.

$$\boldsymbol{B}^{cs}_{k,i} \implies \boldsymbol{q}(t_i) \in \boldsymbol{c}_k \iff \boldsymbol{A}_k \cdot \boldsymbol{q}(t_i) \le \boldsymbol{b}_k, \tag{4.9}$$

where $\boldsymbol{A}_k$ and $\boldsymbol{b}_k$ encodes the geometry of cell $\boldsymbol{c}_k$ using the half-plane representation ($\mathcal{H}$-Rep). The symbol $\iff$ means "if and only if". The following constraint is also imposed

$$\sum_{k=1}^{N_d} \boldsymbol{B}^{cs}_{k,i} = 1, \ \forall i = 1, \cdots, N_t, \tag{4.10}$$

so that at each time, $\boldsymbol{q}$ is assigned to exactly one of the cells.

4.3.2 Mixed-Integer Convex Torque Constraint

In this section, the non-convex torque limit constraint is relaxed into a mixed-integer convex constraint. The torque limit constraint is

$$||\boldsymbol{J}^\top(\boldsymbol{q}) \cdot \boldsymbol{u}||_2 \le \tau_{max} \tag{4.11}$$

where τ_{max} is the maximum actuator torque; $\boldsymbol{q} = [x, z]^\top$ is the robot configuration; $\boldsymbol{u} = [F_x, F_z]^\top$ is the GRF; $||\cdot||_2$ is the two-norm. Given $\boldsymbol{q}$, the joint angles solved from inverse kinematics (IK) is used to construct the Jacobian matrix $\boldsymbol{J}$. The torque limit constraint (4.11) is non-convex because the IK and $\boldsymbol{J}$ calculation involve trigonometric functions. The torque limit constraint (4.11) can be written in the equivalent quadratic form

$$\boldsymbol{u}^\top \boldsymbol{J}(\boldsymbol{q}) \cdot \boldsymbol{J}^\top(\boldsymbol{q})\boldsymbol{u} \leq \tau_{max}^2, \tag{4.12}$$

where $\boldsymbol{J}\boldsymbol{J}^\top$ is a symmetric, positive definite matrix.

The key idea of the proposed mixed-integer convex formulation is imposing a piecewise convex relaxation of the torque constraint over the C-space discretization. The triangle shown in Fig. 4.3(a) is a cell $\boldsymbol{c}_k$ taken from the C-space discretization in Fig. 4.2(b). If the satisfaction of some convex constraint can guarantee that (4.12) holds for every point within the cell $\boldsymbol{c}_k$, then this constraint can be taken as a representative constraint for $\boldsymbol{c}_k$. Suppose N points are sampled from $\boldsymbol{c}_k$, then (4.12) should hold for every sampled points $\boldsymbol{q}_i, i \in \{1, \cdots, N\}$. If there exists a matrix $\boldsymbol{X}$ such that

$$\boldsymbol{u}^\top \boldsymbol{J}(\boldsymbol{q}_i) \cdot \boldsymbol{J}(\boldsymbol{q}_i)^\top \boldsymbol{u} \leq \boldsymbol{u}^\top \boldsymbol{X} \boldsymbol{u} \leq \tau_{max}^2, \qquad \forall i, \tag{4.13}$$

then,

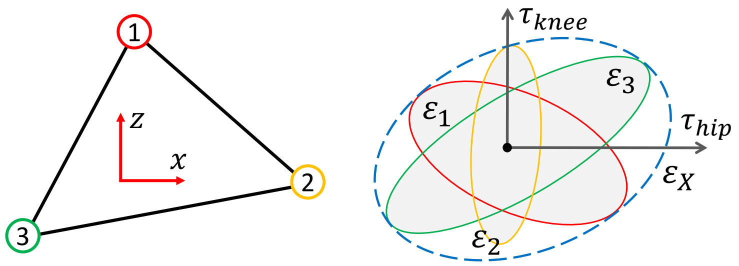
$$\boldsymbol{u}^\top \boldsymbol{X} \boldsymbol{u} \leq \tau_{max}^2 \tag{4.14}$$

is the representative constraint. The matrix $\boldsymbol{X}$ must satisfy the linear matrix inequality (LMI)

$$\boldsymbol{X} \succeq \boldsymbol{J}(\boldsymbol{q}_i)\boldsymbol{J}(\boldsymbol{q}_i)^\top \tag{4.15}$$

for all sampled $\boldsymbol{q}_i$.

The geometric interpretation of (4.13) is presented in Fig.4.3(b). The inequality (4.12) corresponds to an ellipsoid for each sampled $\boldsymbol{q}_i$ in the joint torque space. The red, yellow, and



(a) (b)

Figure 4.3: Geometric interpretation of the minimum bounding ellipsoid problem for the torque limit constraint within a cell. (a) A cell $\boldsymbol{c}_i$ from the C-space discretization (b) $\boldsymbol{\varepsilon}_{1,2,3}$ are the ellipsoids associated with vertices $1, 2, 3$. $\boldsymbol{\varepsilon}_X$ is the minimum bounding ellipsoid.

green ellipsoids in Fig.4.3(b) corresponds to the three vertices of the cell $\boldsymbol{c}_k$ as shown in Fig. 4.3(a). The LMI in (4.15) indicates that the ellipsoid corresponding to $\boldsymbol{X}$ encloses all of the other sampled ones, hence a convex relaxation. To make the relaxation as tight as possible, the volume of the ellipsoid corresponding to $\boldsymbol{X}$ should be minimized. The problem of finding such a minimum bounding ellipsoid can be cast as a semidefinite program (SDP) [15].

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \mathbf{tr}(\boldsymbol{W}\boldsymbol{X}), \\ \text{subject to} \quad & \boldsymbol{X} \succeq \boldsymbol{J}(\boldsymbol{q}_i)\boldsymbol{J}(\boldsymbol{q}_i)^\top, i \in \{1, \cdots, N\}, \end{aligned} \tag{4.16}$$

where $\boldsymbol{X} \in \mathbb{S}_{++}$ is the positive definite optimization matrix variable; $\boldsymbol{J}(\boldsymbol{q}_i)\boldsymbol{J}(\boldsymbol{q}_i)^\top \in \mathbb{S}_{++}$ is the matrix associated with the ellipsoid $\boldsymbol{\varepsilon}_i$. The LMI is valid if and only if $\boldsymbol{\varepsilon}_i \subseteq \boldsymbol{\varepsilon}_X, \forall i \in \{1, \cdots, N\}$. $\boldsymbol{W} \in \mathbb{S}_{++}$ is a scaling matrix, which is set to be identify matrix since the hip and knee motors are the same.

In practice, a finite number of samples are drawn since sampling every point within $\boldsymbol{c}_k$ is not realistic. One key observation is that once the samples increase beyond the vertices, the shape and volume of the minimum-bounding ellipsoid does not change, as shown in Fig. 4.4. Therefore, only the vertices are sampled to solve for the matrix $\boldsymbol{X}$.

One SDP is posed and solved for each cell $\boldsymbol{c}_k$ to find its corresponding $\boldsymbol{X}_k$ so that the

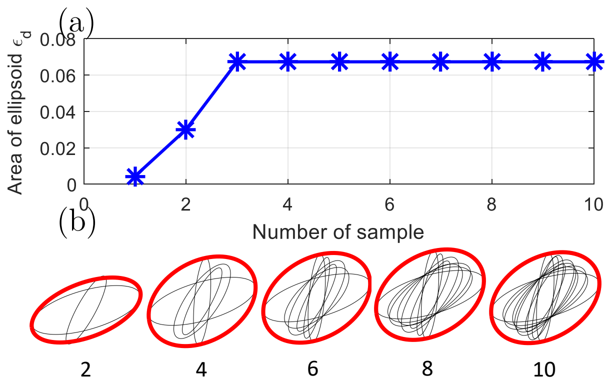


Figure 4.4: The change of minimal outer ellipsoid with respect to the number of sample points taken in one polytope. (a) The trend of volume change (b)The trend of ellipsoid shape change as sample increases.

piecewise convex torque constraint can be imposed

$$\boldsymbol{B}^{cs}_{k,i} \implies \boldsymbol{u}(t_i)^\top \cdot \boldsymbol{X}_k \cdot \boldsymbol{u}(t_i) \leq \tau^2_{max}, \ \forall i, \tag{4.17}$$

where $\boldsymbol{B}^{cs}$ is the binary matrix for C-space discretization as in Section 4.3.1. The constraint (4.17) is quadratic in $\boldsymbol{u}$, which is in affine form of the Bézier polynomial coefficent $\boldsymbol{\alpha}_u$ as introduced in Section 4.2.1. Note that for a given grid resolution, the matrix $\boldsymbol{X}$ only needs to be calculated once. As the discretization resolution (N_d) increases, the convex relaxation of the torque limit constraint becomes less conservative while the computational demand increases.

4.3.3 MIQCP Formulation

The kinodynamic motion planning problem of the R1 robot can be transcribed to a MIQCP, whose optimization variables are

$$\boldsymbol{x}_{opt} = [\boldsymbol{\alpha}_u, \boldsymbol{q}_0, \dot{\boldsymbol{q}}_0, \boldsymbol{T}_{air}, \boldsymbol{p}_{fp}, \boldsymbol{B}^{cs}, \boldsymbol{B}^{fp}, \boldsymbol{B}^{mc}] \tag{4.18}$$

where $\boldsymbol{B}^{cs}, \boldsymbol{B}^{fp}, \boldsymbol{B}^{mc}$ are binary variables for C-space discretization, foothold position and McCormick Envelope, respectively. The complete MIQCP formulation is

$$\underset{\boldsymbol{x}_{opt}}{\text{minimize}} \; f_0(\boldsymbol{x}_{opt}) \tag{4.19a}$$

$$\text{subject to } \boldsymbol{\alpha}_{\dot{q}} = \mathcal{L}(\boldsymbol{\alpha}_u, \dot{\boldsymbol{q}}_0), \boldsymbol{\alpha}_q = \mathcal{L}(\boldsymbol{\alpha}_{\dot{q}}, \boldsymbol{q}_0) \tag{4.19b}$$

$$\boldsymbol{q}(t_0) \in \boldsymbol{Q}_0 \tag{4.19c}$$

$$\boldsymbol{q}(t_f) \in \boldsymbol{Q}_g \tag{4.19d}$$

$$\text{aerial phase kinematics: } (4.5) \tag{4.19e}$$

$$\sum_{s=1}^{N_s} \boldsymbol{B}^{fp}_{s,j} = 1 \tag{4.19f}$$

$$\boldsymbol{B}^{fp}_{s,j} = 1 \implies \boldsymbol{p}^j_{fp} \in seg_s \tag{4.19g}$$

$$\sum_{k=1}^{N_d} \boldsymbol{B}^{cs}_{k,i} = 1 \tag{4.19h}$$

$$\boldsymbol{B}^{cs}_{k,i} \implies \boldsymbol{q}(t_i) \in \boldsymbol{c}_k \tag{4.19i}$$

$$\boldsymbol{B}^{cs}_{k,i} \implies \boldsymbol{u}(t_i)^\top \cdot \boldsymbol{X}_k \cdot \boldsymbol{u}(t_i) \le \tau^2_{max} \tag{4.19j}$$

$$\boldsymbol{u}(t_i) \in \mathcal{C} \tag{4.19k}$$

$$i = 1, \cdots, N_t; j = 1, \cdots, N_j \tag{4.19l}$$

$$k = 1, \cdots, N_d; s = 1, \cdots, N_s \tag{4.19m}$$

where $f_0(\cdot)$ is the convex objective function. For example, f_0 can be $||\boldsymbol{q}(t_f) - \boldsymbol{q}_g||_2$, where $\boldsymbol{q}_g$ is the goal configuration. Constraint (4.19b) encodes the simplified dynamics (4.2) as linear relationships of Bézier polynomial coefficients; (4.19c) and (4.19d) restrains the initial and final configurations within their corresponding sets $\boldsymbol{Q}_0$ and $\boldsymbol{Q}_g$, respectively. The aerial phase kinematics involves the binary matrix $\boldsymbol{B}^{mc}$; (4.19f) and ((4.19g)) are mixed-integer affine constraints for foothold position choice; (4.19h) - (4.19j) impose the mixed-integer convex quadratic relaxed torque limit constraint, whose original non-convex control constraint is (4.1g). To prevent contact foot from slipping, GRF are constrained within the friction cone

$\mathcal{C}$ defined as

$$\mathcal{C} = \{\boldsymbol{u} | \ \ |u_t| \leq \mu u_n \geq 0\} \tag{4.20}$$

where the subscripts $(\cdot)_t$ and $(\cdot)_n$ refers to tangential and normal components of the GRF; μ is the friction coefficient.

With the objective function of (4.19) being convex, (4.19j) being a (mixed-integer) quadratic constraint, and the rest being (mixed-integer) affine in $\boldsymbol{x}_{opt}$, the mathematical program (4.19) is a mixed-integer quadratically-constrained convex program (MIQCP).

The MIQCP (4.19) is formulated in MATLAB using YALMIP [83]. The SDP is solved using the solver MOSEK [97] and the MIQCP is solved using the solver CPLEX [63] on a desktop with 2.9 GHz Intel i7.

4.3.4 Results

This section presents the application results of the kinodynamic motion planning algorithm on the R1 robot. Specifically, a Parkour motion of two consecutive jumps is found by the MIQCP without any initial guess, which is one of the benefits of the MIP formulation. The validity of the solved trajectories is proved by hardware experiment results produced by the R1 robot.

Experiment Setup

The R1 robot is a 2 DoF leg module rigidly mounted on the end of a passive boom system with radius $R_{boom} = 1.25$ m. The R1 robot is 0.91 kg and its hip and knee joints are equipped with an encoder. Off-board power source and computer are tethered to the robot. Two encoders are installed on the base of the boom to measure the CoM position of the robot. The feed-forward force obtained by solving the MIQCP off-line was applied during the stance phase. An operational-space PD controller was used to track swing foot trajectory during the flight phase. Contact detection as described in Section 2.4.3 was implemented to

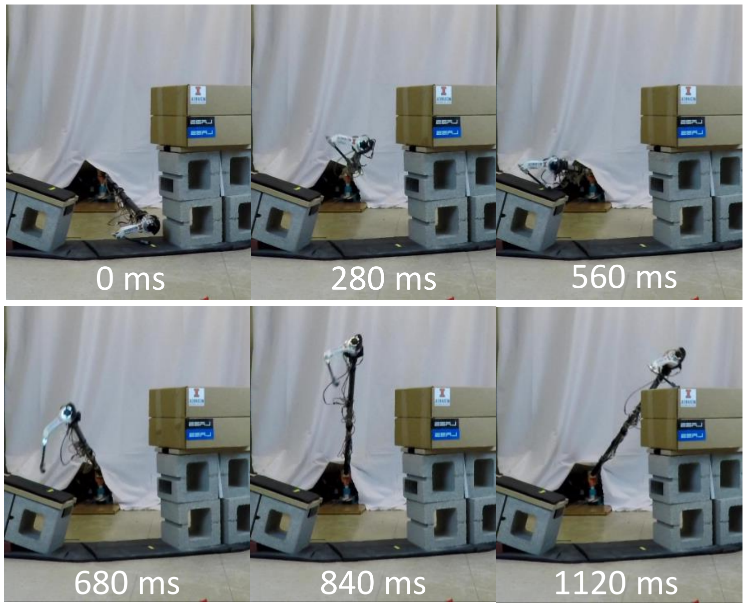


Figure 4.5: Sequential snapshots of the Parkour experiment where the robot makes two consecutive jumps to traverse terrain with large height variation. The MIQCP-based planner is able to find the solution that utilizes the left obstacle as a stepping stone to reach the 0.66 m high goal region.

initiate the stance phase.

Experiment Result

The MIQCP dynamic motion planning algorithm is used to solve for the jumping trajectory that can overcome a terrain shown in Fig. 4.5. The surface of the left platform is sloped, which prohibits taking individual single jumps since the robot cannot stand statically on the slope.

The R1 robot is tasked to reach the goal region on the right obstacle, which cannot be reached with a single jump due to the torque limit. The proposed MIQCP algorithm can find the strategy that utilizes the sloped obstacle on the left as a stepping stone towards the goal region, as shown in Fig. 4.5. Note that R1 adopted the strategy of making two consecutive jumps to overcome the sloped surface on the left platform.

Fig. 4.6 shows the experiment CoM trajectory closely follows the simulation CoM trajectory. The deviation is in part caused by the foot swing movement, which is shown as the

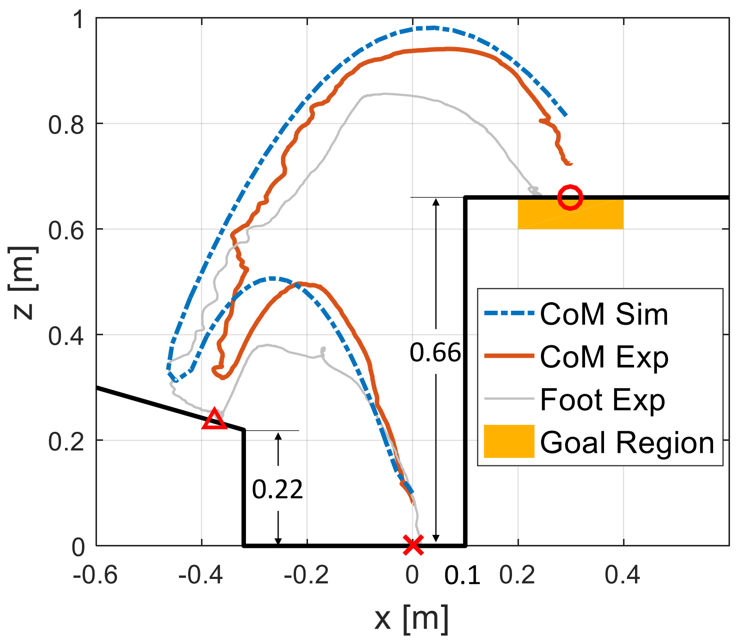


Figure 4.6: The CoM trajectories from the MIQCP-based planner (blue) and the experimental platform (red). The swing leg motions cause the CoM

gray curve. This problem could be solved within 0.9 seconds.

4.4 Planar Two-Legged Robot R2

This section presents how the MICP-based kinodynamic motion planning framework used for R1 can be naturally extended to R2. The R2 robot is more complex due to the additional leg and torso pitch DoF, which increases the state dimension and complicates the contact scenario. Specifically, unlike the unique stance mode of R1, R2 can take a single (under-actuated) or double stance (over-actuated). To address these problems, a more general polytopic wrench constraint takes place of the ellipsoidal torque limit constraint in Section 4.3.2, making the final formulation a mixed-integer convex program (MICP). Specifically, the input to the system changes from the GRF at the contact foot to the effective body wrench generated by the GRF. This choice of input provides a more general framework that incorporates different contact scenarios.

4.4.1 C-Space Discretization

A schematics of the R2 robot is shown in Fig. 4.7(a), where $\{S\}$ refers to the world frame and $\{B\}$ the local frame. The C-space shown in Fig.4.7(b) is divided into three disjoint regions corresponding to front stance $\boldsymbol{\Omega}_{fs}$ (blue), double stance $\boldsymbol{\Omega}_{ds}$ (black), and back stance $\boldsymbol{\Omega}_{bs}$ (red), namely,

$$\boldsymbol{\Omega} = \boldsymbol{\Omega}_{bs} \cup \boldsymbol{\Omega}_{fs} \cup \boldsymbol{\Omega}_{ds}. \tag{4.21}$$

Such a construction of C-space is based on the following assumptions:

Assumption 1 (Fixed Stance Width)**.** *When in double stance, the distance between two contact feet is equal to the body length.*

This assumption enables the C-space to be fixed given the body length and leg linkage length. Furthermore, all quantities in (4.4) can be retrieved through inverse and forward kinematic calculation based on the robot configuration $\boldsymbol{q}$. Fig. 4.7(b) shows the C-space of the robot with parameters in Table 2.1.

When double stance is kinematically feasible, so are the front and back stance. To simplify the choice of stance mode, the following assumption is made.

Assumption 2 (Preference on Double Stance)**.** *When possible, the R2 robot prefers double stance to single stance.*

The basis of assumption 2 is the observation that a double stance provides more control authority compared with a single stance. Assumption 2 establishes a one-to-one mapping between robot configuration $\boldsymbol{q}$ and stance mode, as shown in Fig. 4.7 (c). Namely, the contact mode scheduling automatically emerges as the trajectory $\boldsymbol{q}(t)$ is solved.

Similar to Section 4.3.1, the non-convex C-space of R2 is discretized into cells of tetrahedrons. The discretization is arranged such that each tetrahedron resides within the same stance region. Each tetrahedron is defined using a set of linear inequality constraints

$$\boldsymbol{c}_k := \{\boldsymbol{q} \in \mathbb{R}^3 \mid \boldsymbol{A}_k \cdot \boldsymbol{q} \leq \boldsymbol{b}_k\}, \forall k = 1, \cdots, N_d. \tag{4.22}$$

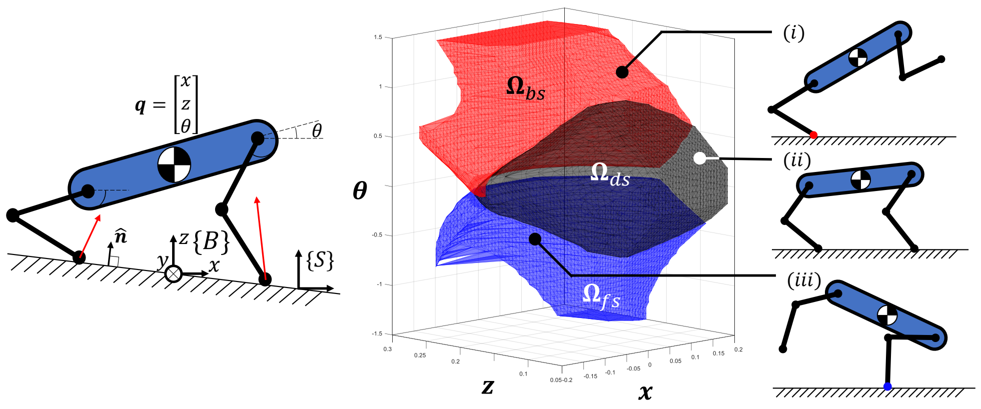


Figure 4.7: The schematics and C-space segmentation of R2. (a) The schematics of R2. The coordinate $\{S\}$ refers to the world frame and $\{B\}$ the local frame. (b) The C-space of R2, which is divided into three mutually exclusive regions corresponding to back stance (red), double stance (black), and front stance (blue), respectively. (c) The illustration of robot configuration in each corresponding stance mode.

4.4.2 Feasible Wrench Polytope

This section presents a formulation of the feasible wrench polytope (FWP) that is pertinent to the robot system studied in this section. A more comprehensive derivation of the FWP can be found in [104].

The feasible force polytope (FFP) for one leg is defined as

$$FFP := \{\boldsymbol{f} \in \mathbb{R}^2 \mid |\boldsymbol{J}^\top \boldsymbol{f}|_\infty \le \tau_{max} \tag{4.23a}$$

$$|\boldsymbol{f} - \hat{\boldsymbol{n}}^\top \boldsymbol{f}|_\infty \le \mu \hat{\boldsymbol{n}}^\top \boldsymbol{f}\}, \tag{4.23b}$$

where $\boldsymbol{J}$ is the Jacobian matrix; τ_{max} is the joint torque limit; μ is the coefficient of friction. The inequality (4.23a) encodes the joint torque constraint $|\boldsymbol{\tau}|_\infty < \tau_{max}$ using the infinity norm. The inequality (4.23b) represents the friction cone constraint. The FFP is a function of the robot configuration $\boldsymbol{q}$ because of assumption 1. An example of FFP is shown in Fig. 4.8(a), where the FFP of front leg is colored blue and that of back leg colored red.

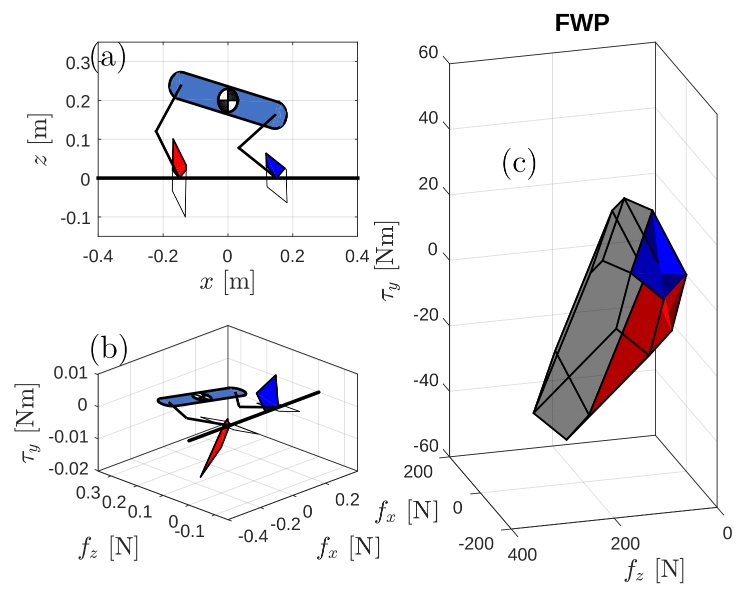


Figure 4.8: FWP visualization of the R2 robot. (a) The FFP of front (blue) and back (red) legs. (b) The FWP of both legs plotted in the 3D wrench space. (c) The FWP_q of the robot is the Minkowski sum of the FWP of both legs.

The FWP of one leg is the set of wrench that can be produced by the FFP

$$FWP_i := \{\boldsymbol{u}_i \in \mathbb{R}^3 \mid \boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{f}_i^k \\ \boldsymbol{r}_i \wedge \boldsymbol{f}_i^k \end{bmatrix}, \boldsymbol{f}_i^k \in FFP_i\}, \tag{4.24}$$

where $\boldsymbol{f}_i^k$ is the k^{th} vertex of FFP for contact point i. Note that the FWP is defined using the vertex-representation of a polytope ($\mathcal{V}$-Rep). Fig. 4.8(b) shows the FWP of front leg (blue) and back leg (red), which are 2-D polytopes embedded in the 3-D wrench space.

When R2 is in double stance, the effective FWP about the CoM is the Minkowsi sum [135] of the FWP created by each contact foot,

$$FWP_{ds} = \bigoplus_{i=1}^{2} FWP_i, \tag{4.25}$$

where FWP_{ds} indicates the double stance FWP. The Minkowski sum of two sets X and Y is $X \oplus Y := \{x + y | x \in X, y \in Y\}$. Fig. 4.8(c) shows an example of FWP_{ds} as the Minkowski sum of the FWP of both legs. Assumption 2 implies that when $\boldsymbol{q} \in \boldsymbol{\Omega}_{ds}$, the corresponding FWP_{ds} is defined as in (4.25).

For a polytope $\boldsymbol{c}_k$ in the C-space discretization, its representative FWP is

$$FWP_{c_k} = \begin{cases} \cap_{v=1}^{4} FWP_{c_k,v}, & \text{double stance} \\ FWP_{c_k}^{cbsv}, & \text{single stance,} \end{cases} \tag{4.26}$$

where $FWP_{c_k,v}$ is the FWP at vertex v^{th} of the cell $\boldsymbol{c}_k$. $FWP_{c_k}^{cbsv}$ is the FWP at the Chebyshev center [15], which is the center of the largest Euclidean ball that lies in a polytope. The choice of FWP for double stance in (4.26) is conservative because it is the intersection of the FWP at all of the 4 vertices of the cell, hence providing robustness when the robot is in the double stance. In comparison, the FWP at each vertex of a single stance cell degenerates to a 2-D polytope due to the coupling between forces and moment. Since each vertex corresponds to a different $\boldsymbol{r}_i$, the FWP of a single stance cell have no intersection except the origin. Therefore, the FWP for a single stance cell is defined at the Chebyshev center of the cell.

The FWP_{c_k} can be represented using the half-plane representation ($\mathcal{H}$-Rep) consisting of a set of linear constraints

$$FWP_{c_k} := \{\boldsymbol{u} \in \mathbb{R}^3 \mid \boldsymbol{A}_k^{fwp} \cdot \boldsymbol{u} \leq \boldsymbol{b}_k^{fwp}\}, k = 1, \cdots, N_d, \tag{4.27}$$

where $\boldsymbol{A}_k^{fwp}$ and $\boldsymbol{b}_k^{fwp}$ encode the geometry of the FWP at cell $\boldsymbol{c}_k$. The single stance FWP is subject to equality constraints, which can also be incorporated into the form of inequality constraint as in (4.27).

Similar to the matrix $\boldsymbol{X}$ in Section 4.3.2, the FWP_{c_k} only needs to be computed once for a given set of robot physical parameters and C-space discretization.

4.4.3 Mixed-Integer Wrench Constraint

The non-convex wrench constraint is imposed in a piecewise constant fashion over the discretized C-space, similar to the mixed-integer convex torque limit constraint presented in Section 4.3.2. A binary matrix $\boldsymbol{B}^{cs} \in \{0,1\}^{N_t \times N_d}$ is constructed such that $\boldsymbol{B}^{cs}_{k,i} = 1$ indicates that $\boldsymbol{q}(t_i)$ is within cell $\boldsymbol{c}_k$ and the spatial wrench should be chosen within FWP_{c_k}

$$\begin{aligned} \boldsymbol{B}^{cs}_{k,i} &\implies \boldsymbol{q}(t_i) \in \boldsymbol{c}_k \\ &\implies \boldsymbol{u}(t_i) \in FWP_{c_k}, \end{aligned} \tag{4.28}$$

where the *implies* operator ($\implies$) is implemented using the big-M formulation [119]. Additional constraints

$$\sum_{k=1}^{N_d} \boldsymbol{B}^{cs}_{k,i} = 1,\ \forall i = 1, \cdots, N_t, \tag{4.29}$$

are imposed so that at each time t_i, $\boldsymbol{q}(t_i)$ resides within exactly one cell.

4.4.4 MICP Formulation

Similar to the MIQCP presented in Section 4.3.3, the kinodynamic motion planning problem of R2 can be transcribed to a MICP with mixed-integer affine constraints. The optimization variables are

$$\boldsymbol{x}_{opt} = [\boldsymbol{\alpha}_u, \boldsymbol{q}_0, \dot{\boldsymbol{q}}_0, \boldsymbol{T}_{air}, \boldsymbol{p}_{fp}, \boldsymbol{B}^{cs}, \boldsymbol{B}^{fp}, \boldsymbol{B}^{mc}], \tag{4.30}$$

where the definition of each entry in $\boldsymbol{x}_{opt}$ is identical to that in (4.3.3). The complete MICP formulation is:

$$\underset{\boldsymbol{x}_{opt}}{\text{minimize}} \; f_0(\boldsymbol{x}_{opt}) \tag{4.31a}$$

$$\text{subject to } \boldsymbol{\alpha}_{\dot{q}} = \mathcal{L}(\boldsymbol{\alpha}_u, \dot{\boldsymbol{q}}_0), \boldsymbol{\alpha}_q = \mathcal{L}(\boldsymbol{\alpha}_{\dot{q}}, \boldsymbol{q}_0) \tag{4.31b}$$

$$\boldsymbol{q}(t_0) \in \boldsymbol{Q}_0 \tag{4.31c}$$

$$\boldsymbol{q}(t_f) \in \boldsymbol{Q}_g \tag{4.31d}$$

$$\text{aerial phase kinematics: (4.5)} \tag{4.31e}$$

$$\sum_{s=1}^{N_s} \boldsymbol{B}^{fp}_{s,j} = 1 \tag{4.31f}$$

$$\boldsymbol{B}^{fp}_{s,j} = 1 \implies \boldsymbol{p}^{j}_{fp} \in seg_s \tag{4.31g}$$

$$\sum_{k=1}^{N_d} \boldsymbol{B}^{cs}_{k,i} = 1 \tag{4.31h}$$

$$\boldsymbol{B}^{cs}_{k,i} \implies \boldsymbol{q}(t_i) \in \boldsymbol{c}_k \tag{4.31i}$$

$$\boldsymbol{B}^{cs}_{k,i} \implies \boldsymbol{u}(t_i) \in FWP_{c_k} \tag{4.31j}$$

$$i = 1, \cdots, N_t; j = 1, \cdots, N_j \tag{4.31k}$$

$$k = 1, \cdots, N_d; s = 1, \cdots, N_s. \tag{4.31l}$$

The terms are defined similarly to that of MIQCP (4.19) except constraint (4.31j), where the FWP constraint replaces the quadratic torque constraint (4.19j). Note that there is no friction cone constraint since it has been incorporated in FWP. The convex objective $f_0(\cdot)$ is a task-specific function chosen by design. For example, f_0 can be $||\boldsymbol{q}(t_f) - \boldsymbol{q}_g||$, which makes the problem a mixed-integer quadratic program (MIQP). f_0 can also be $-\boldsymbol{q}_x(t_f)$ to maximize horizontal jumping distance, which leads to a mixed-integer linear program (MILP); the objective can also be set as a constant value to solve a feasibility problem.

The MICP problem is formulated in MATLAB using YALMIP [83]. The computational geometry calculation related to FWP is done using the Multi-Parametric Toolbox 3 (MPT3)

[52]. The MICP is solved by the solver Gurobi [102]. All of the computation is performed on a desktop with 2.9 GHz Intel i7.

4.4.5 Results

To validate the proposed kinodynamic motion planning algorithm, jumping experiments are conducted on the robot. Experiment results for both jumping forward and backward, together with the simulation result of a dynamic Parkour motion are presented. Note that the trajectories of all three motions are solved by the proposed MICP without any initial guesses.

Experimental Setup

The experimental setup of the R2 robot is shown in Fig. 4.9. Similar to the setup used in [81], the robot is composed of a torso made of a carbon fiber tube and two legs modules. An inertial measurement unit (IMU) is mounted on the torso for state estimation. The center of the torso is connected via a bearing to the end of the boom system identical to the one used in Section 4.3.4. An encoder is mounted at the connection between the tip of the boom and the robot to measure the pitch angle θ. The total mass of the robot is 2.56 kg, and the rest of the physical parameters can be found in Table 2.1.

Jump On Platforms

A picture where the robot jumps forward and upward onto a 0.2 m high platform (80% of robot height) is shown in Fig. 4.9. Another experiment where the robot jumps back onto the platform is shown in Fig. 4.10. It can be observed that the motion involves large body pitch oscillation that steers the swing foot clear of the obstacle and aids the robot to accomplish the task. For both motions, the robot configuration is constrained within the double stance region $\boldsymbol{\Omega}_{ds}$ to accelerate the computation. The wrench trajectory $\boldsymbol{u}(t)$ obtained from solving the MICP is distributed to the GRF using the closed-chain-constrained operational-space

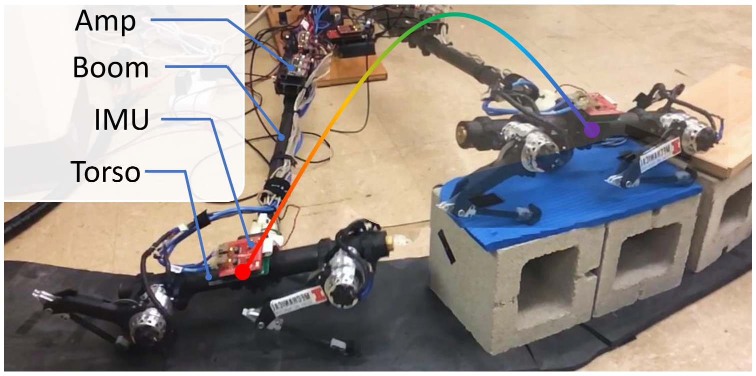


Figure 4.9: The hardware experiment where the two-legged planar robot executed a dynamic motion generated by the MIQP and mounted an obstacle 80% of its height. The robot is mounted on a boom system to constrain its motion within the sagittal plane.

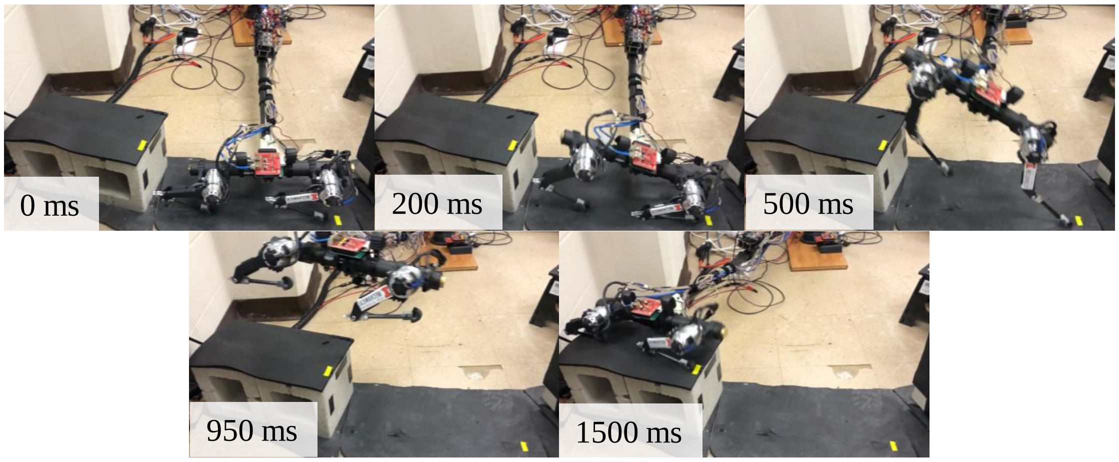


Figure 4.10: Sequential snapshots of the experiment where the robot executed the dynamic motion generated by the MIQP and jumped backward onto a 0.2 m high platform.

control [61], similar to the frontal plane controller in [107]. The number of variables for both motions is 216 (27 continuous, 189 integers), and the computational time to solve the MICP is 0.84 s for the jumping forward problem and 5.94 s for the jumping backward problem. The solve time difference may be explained by that the knee-bending-back configuration provides more forward force authority. Additionally, the knee-bending-forward configuration imposes stricter collision avoidance constraints between the knee and terrain. The Bézier coefficients of the wrench trajectories are summarized in Table 4.2.

Experiment	Wrench	Bézier Coefficients
Jump forward	$\boldsymbol{\alpha}_{f_x}$	0.0, -7.6, 33.8, -33.7, 90.1, 0.0
	$\boldsymbol{\alpha}_{f_z}$	25.1, -69.0, 152.0, -50.0, 262.7, 0.0
	$\boldsymbol{\alpha}_{\tau_y}$	0.0, 5.5, -25.3, 25.5, -5.8, 0.0
Jump backward	$\boldsymbol{\alpha}_{f_x}$	0.0, 198.4, -404.8, 296.7, -135.5, 0.0
	$\boldsymbol{\alpha}_{f_z}$	25.1, -171.9, 631.8, -786.8, 623.5, 0.0
	$\boldsymbol{\alpha}_{\tau_y}$	0.0, 44.5, -79.4, 48.2, -14.5, 0.0

Table 4.2: Bézier coefficient for the jumping on platform experiments

Parkour Motion

The proposed kinodynamic motion planning framework can provide complex maneuver plans to traverse challenging terrains. For example, Fig. 4.11 shows a terrain where the robot cannot reach the goal region on the high platform with a single jump due to actuator limitations. The solution that MICP provides is a Parkour motion that exploits the left platform as a stepping stone towards the goal region by making two consecutive jumps. In addition, the proposed MICP approach solves the problem without initial guess nor user input about the step planning. With the grid resolution $N_{bs} = 10, N_{fs} = 10, N_{ds} = 21$, the MICP involves 485 variables (53 continuous, 432 integer), and the computational time is 27 s. The robot is in a back stance towards the end of the second jump, presumably to exploit the body pitch for extra kinematic reachability. This simulation result of Parkour motion showcases one of the advantages of MICP-based motion planning algorithms, which is that it can reason about making discrete decisions.

4.5 Summary

This chapter presents a mixed-integer convex programs (MICP)-based kinodynamic motion planning framework for dynamic legged robots to traverse challenging terrains. This terrain traversal problem could be posed as a trajectory optimization (TO) problem. However, the non-convex torque limit constraint often induces solutions trapped in local minima, depending on the initial guess. The novel MICP-based planning framework can provide a certificate

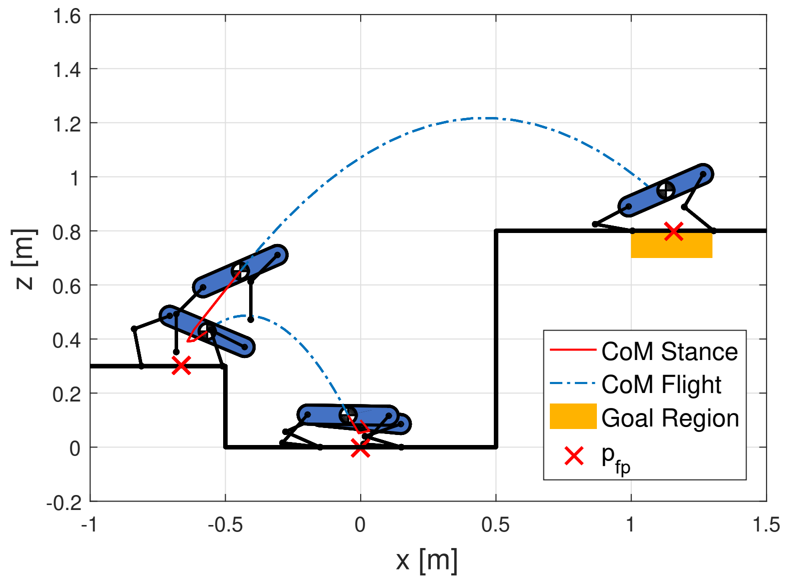


Figure 4.11: Simulation result of the Parkour motion. The proposed formulation can find the strategy of utilizing the left platform as a stepping-stone to reach the goal region on the high platform.

for global optimality or infeasibility, while not requiring an initial trajectory. Specifically, the non-convex control constraint is replaced by piece-wise convex relaxation over the C-space discretization using the mixed-integer formulation. Other non-convex constraints such as bilinear terms and foothold position choice can also fit into the mixed-integer framework, which makes the final problem a MICP. Dynamically feasible trajectories can be obtained by off-the-shelf numerical solvers, which can solve the MICP efficiently to global optimality given the discretization resolution. Experiments on two robot platforms R1 and R2 showcase the efficacy of the proposed framework to enable dynamic legged robots to overcome challenging terrains.

One of the applications of the MICP-based planner is to automatically generate initial guesses to a full trajectory optimization with higher model fidelity. Another potential application is dynamic TO for manipulators. The major drawback of this scheme, however, is that the exponentially growing solve time as the number of decision variable increases. This disadvantage prohibits the application of the MICP-based method to higher dimensional space. Nevertheless, advancements in computer hardware and efficient algorithm, such as the recent results on warm-starting a B&B solver [84], can potentially mitigate this problem.

Chapter 5

Hybrid Sample/Optimization - based Planning for Jumping Robots

This chapter proposes a hybrid planning framework that generates long-horizon dynamic motion plans for jumping legged robots to overcome complex terrains. By employing a motion primitive [48], which is a pre-defined and pre-computed motion, the original problem is decoupled as path planning followed by a trajectory optimization (TO) module that handles dynamics. A variant of a kinodynamic Rapidly-exploring Random Trees (RRT) planner finds a path as a parabola sequence between stance phases. To accelerate the computation process, a reachability informed control sampling scheme is proposed to leverage the pre-computed velocity reachability map. The path is post-processed to eliminate redundant jumps and passed to the TO module to find a dynamically feasible trajectory. Simulation results are presented where the proposed hybrid planner navigates a single-legged robot through complex obstacles by executing multiple consecutive jumps, producing novel strategies to leap over large gaps by leveraging dynamics. In a physical experiment, the hybrid planner is tested on a real robot successfully traversing challenging terrain. This chapter is based on our work in [35] [1].

5.1 Introduction

Legged systems have the unique capability of making jumps to overcome challenging terrains with large height differences and wide gaps. Agile animals such as squirrels can plan complex dynamic maneuvers that fully utilize their inherent dynamics to jump over extremely difficult

[1]Video clips featuring the simulation and experiment are available in movie1

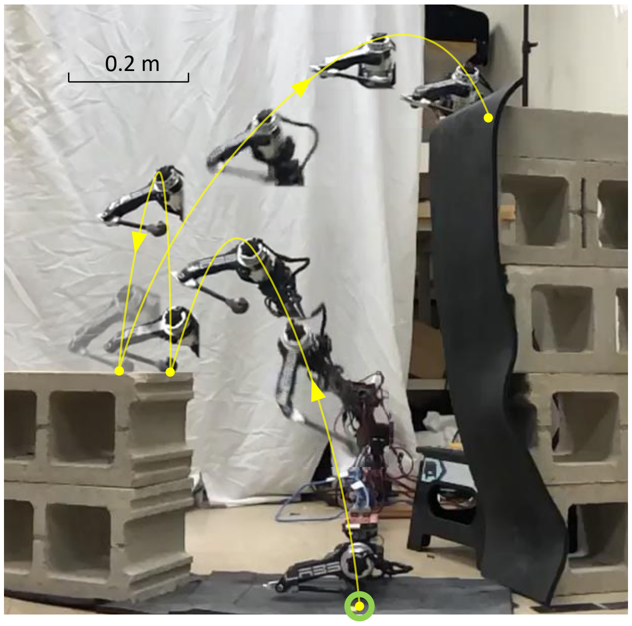


Figure 5.1: Snapshots of the experiment where the robot executed three consecutive jumps produced by the proposed hybrid planner and surmounted the 0.9 m high platform. The R1 robot is mounted on a boom system.

obstacle tracks. To rival nimble animals, many legged robots with high jumping capability have been developed [46, 62, 67, 153]. However, due to the differential constraints imposed by the robot dynamics and the hybrid nature inherent to locomotion, motion planning algorithms that can enable these jumping robots to traverse complex terrains are not as well developed.

Campana and Laumond proposed a ballistic motion planning algorithm [19] that can find a path with friction cone and velocity constraints in a complex 3D environment using Probabilistic Roadmaps (PRM) [68]. Inspired by [19], this work aims to improve upon the assumption of impulsive stance phase by explicitly addressing the stance dynamics, so that more innovative trajectory can be discovered. To achieve this goal, this work adopts the kinodynamic motion planning [38] view point, which respects the dynamics of the robot by imposing it as differential constraints. Sampling-based methods such as PRM and Rapidly-exploring Random Trees (RRT) [77] are widely used to solve large planning problems. However, the presence of certain differential constraints can severely compromise the efficiency of these algorithms. Reachability-guided RRT (RG-RRT) [127] increases the sampling efficiency by taking into consideration of local reachability. Implementing RG-RRT in task space and utilizing motion primitives enabled the LittleDog to bound over rough terrain [126]. To increase the efficiency of kinodynamic planners, Bézier curves [76] are used since they parametrize a trajectory with fewer parameters. On the other extreme, solving the whole trajectory with full dynamics results in a large trajectory optimization (TO) problem. TO has been widely used to generate dynamic motion plans for humanoids [24, 75] and quadrupeds [67, 109] since it handles the state and control constraints in a nonlinear program (NLP) formulation. However, the computation time increases drastically as the planning horizon increases.

To exploit the advantages of both sampling and optimization-based methods, this paper proposes a hybrid sampling/optimization-based planner for generating dynamic motions for single-legged jumping robots to traverse challenging terrains. We decouple the original problem into sampling-based planning followed by a module that solves for the full dynamics using optimization. Similar to [19], aerial phases are constructed as parabolas connected by stance phases. Since the relationship between the touchdown and liftoff state is complex, a velocity reachability map is pre-computed and used in the kinodynamic RRT. After a feasible path is found by the kinodynamic RRT and post-processed, trajectory optimization is performed at each stance to find the state and control trajectories.

The proposed hybrid planner is benchmarked against a quasi-static planner and a mixed-integer convex program (MICP) based planner. Compared with the quasi-static planner, our method is momentum aware in the sense that it can find strategies where the robot makes consecutive jumps to gain momentum to clear a wide gap. Compared with the MICP-based planner, the solve time of the hybrid planner scales much better as the number of step increases. To validate the trajectory produced by the hybrid planner, a physical experiment is conducted on robot hardware and snapshots of the experiment are presented in Fig. 5.1.

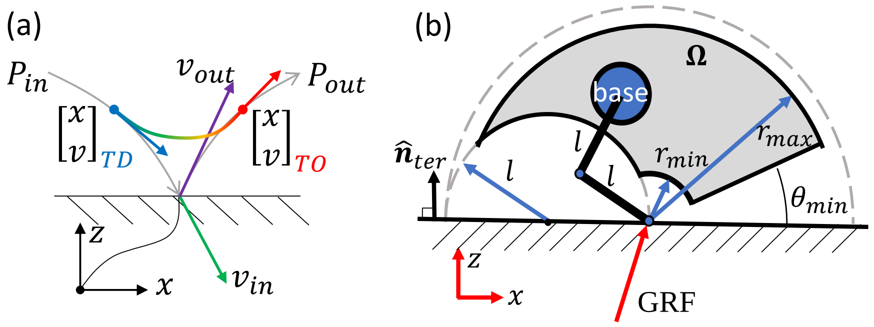

(a)
P_{in}
v_{out}
P_{out}
$\begin{bmatrix} x \\ v \end{bmatrix}_{TD}$
$\begin{bmatrix} x \\ v \end{bmatrix}_{TO}$
z
x
v_{in}
(b)
Ω
base
$\hat{\boldsymbol{n}}_{ter}$
l
l
l
r_{min}
r_{max}
θ_{min}
z
x
GRF

Figure 5.2: (a) Illustration of the motion primitive. To connect two neighboring parabolas, the boundary states of stance are constrained on their corresponding parabola (gray curve). (b) The schematic of the single-legged robot when it is in stance phase. The configuration Ω is the gray shaded area.

5.2 Hybrid Planning Pre-requisites

The hybrid planning approach decomposes the complex kinodynamic motion planning problem into two stages. The first stage utilizes a variant of kinodynamic RRT to find a sequence of parabolas that connects the start and the goal. The second stage applies TO at each stance to solve for the full dynamics. The motion primitive [48] that enables the hybrid framework is shown in Fig. 5.2. The incoming parabola $\boldsymbol{P}_{in}$ and outgoing parabola $\boldsymbol{P}_{out}$ are parametrized by $\boldsymbol{v}_{in}$ and $\boldsymbol{v}_{out}$, respectively. Note that these are variables that only pertain to the kinematic path, which is generated by the sampling-based planning stage. The touchdown states of the TO $\boldsymbol{x}_{TD}$ and the liftoff state $\boldsymbol{x}_{LO}$ are chosen on $\boldsymbol{P}_{in}$ and $\boldsymbol{P}_{out}$, respectively. Therefore, the aerial phase determined by the first stage is preserved in the second stage. The added benefit is that since each stance is isolated, the multiple TOs can be parallelized.

If TO fails in the second stage, the failed jump is removed from the tree, and sampling-based planning resumes to generate more TO candidates. We observe that failures are infrequent due to our use of a *velocity reachability map* (defined in Section 5.2.2) within sampling-based planning to generate feasible jumps with high likelihood.

5.2.1 Stance Trajectory Existence

A fundamental subproblem in our approach is a boundary value problem to determine whether a feasible trajectory at a stance can connect prescribed incoming and outgoing states. It can be formulated as a trajectory optimization problem, called TO^1, as follows:

$$\begin{aligned}
\underset{\boldsymbol{\alpha}_F, T_{st}}{\text{minimize}} \quad & \sum_{k=1}^{N} ||\boldsymbol{\tau}_k|| \cdot T_{st} && \text{(5.1a)}\\
\text{subject to} \quad & \boldsymbol{p}_k \in \Omega && \text{(5.1b)}\\
& \tau_{min} \leq \boldsymbol{\tau}_k \leq \tau_{max} && \text{(5.1c)}\\
& \boldsymbol{x}_{TD} \in \boldsymbol{P}_{in}, \boldsymbol{x}_{LO} \in \boldsymbol{P}_{out} && \text{(5.1d)}\\
& \boldsymbol{F}_k \in \mathcal{C}(\mu). && \text{(5.1e)}
\end{aligned}$$

Since the robot starts and ends at the static pose at the first and last jump, the initial position of the first jump and the final position of the last jump is only subject to the workspace constraint.

To make the problem finite-dimensional, the state and control trajectories are discretized at N sample points and subscript $(\cdot)_k$ indicates values at the k^{th} instance of the sampled time. The resulting optimization problem is a nonlinear program (NLP).

5.2.2 Velocity Reachability Map

Given an incoming velocity $\boldsymbol{v}_{in}$ at a stance, the velocity reachability map $\mathcal{R}(\boldsymbol{v}_{in})$ is defined as the set of $\boldsymbol{v}_{out}$ such that that $TO^1(\boldsymbol{v}_{in}, \boldsymbol{v}_{out})$ in (5.1) has a solution. The reverse reachability map $\mathcal{R}^{-1}(\boldsymbol{v}_{out})$ is defined similarly.

For a given single-legged robot, we precompute an approximation of the reachability map that is used in the sampling-based planner to greatly speed up planning by limiting connections so that they have a high probability of yielding a dynamically feasible trajectory. We approximate $\mathcal{R}(\boldsymbol{v}_{in})$ by running TO^1 over a 4D grid of $\boldsymbol{v}_{in}$, $\boldsymbol{v}_{out}$ and recording successes and failures. Even though an NLP can be trapped in local minima, TO^1 works sufficiently well for the small-scale problem as is the case here. For each $\boldsymbol{v}_{in}$, the successful $\boldsymbol{v}_{out}$ are approximated with a convex hull, which may under-approximate the true reachability at the margins but over-approximate it in convex regions. The same dataset is used to derive an approximation of $\mathcal{R}^{-1}$ in a similar fashion.

Fig. 5.3 presents a illustration of the velocity reachability map. The bottom area is the set of $\boldsymbol{v}_{in}$ with non-empty $\mathcal{R}(\boldsymbol{v}_{in})$. Each cell representing a $\boldsymbol{v}_{in}$ color coded by the area covered by $\mathcal{R}(\boldsymbol{v}_{in})$. Note that the set of valid $\boldsymbol{v}_{in}$ is asymmetric because the serial linkage leg of the robot bends towards one side. The $\mathcal{R}$ of two sample $\boldsymbol{v}_{in}$ are plotted on the top of Fig. 5.3, where the cross symbol represents that TO^1 can find a solution for the $\boldsymbol{v}_{in}, \boldsymbol{v}_{out}$ pair. The set $\mathcal{R}(\boldsymbol{v}_{in})$ is defined as the convex hull of the crossed points.

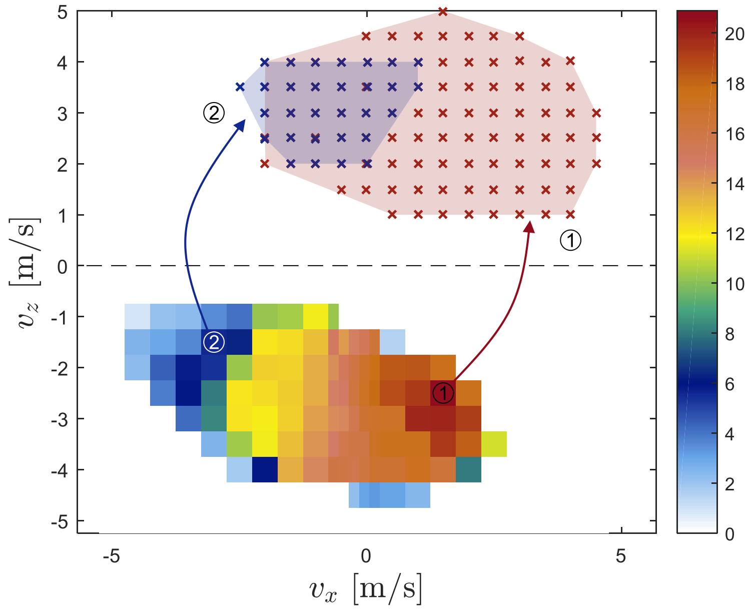


Figure 5.3: An illustration of the reachability map $\mathcal{R}$. The bottom area is the set of $\boldsymbol{v}_{in}$ with a non-empty $\boldsymbol{v}_{out}$ set; the color at each $\boldsymbol{v}_{in}$ indicates the total area of the corresponding $\boldsymbol{v}_{out}$ set. Two sample $\boldsymbol{v}_{out}$ sets are shown at the top.

5.3 Hybrid Sampling/Optimization-based Plannng Algorithms

5.3.1 Sampling-Based Planning

Sampling-based planning forms the outer loop of the hybrid motion planner as shown in Algorithm 1. We use a variant of the kinodynamic RRT algorithm, modified with a reachability-informed control sampling scheme.

Algorithm 1 `Hybrid Motion Planner`

```
Input: $\mathcal{R}, \boldsymbol{x}_0, \boldsymbol{x}_g, Terrain$
Output: $Traj$                                         ▷ Jumping trajectory
 1: $\mathcal{T}.init(\boldsymbol{x}_0)$
 2: $finished = False$
 3: while not $finished$ do
 4:     $\boldsymbol{x}_{rand} \leftarrow RandomSample(Terrain)$
 5:     $\boldsymbol{x}_{parent} \leftarrow FindParent(\mathcal{T}, \boldsymbol{x}_{rand})$
 6:     $\boldsymbol{x}_{new} \leftarrow STEER(\mathcal{R}, \boldsymbol{x}_{rand}, \boldsymbol{x}_{parent}, Terrain)$
 7:     if $\boldsymbol{x}_{new} \neq null$ then
 8:         $\mathcal{T}.add(\boldsymbol{x}_{parent} \rightarrow \boldsymbol{x}_{new})$
 9:         if $ReachGoal(\boldsymbol{x}_{new})$ then
10:             $path = \mathcal{T}.FindPath()$
11:             $path^* = path.Shortcut()$
12:             $\boldsymbol{x}_{fail}, traj = TrajOpt(path^*)$
13:             if $\boldsymbol{x}_{fail} = null$ then return $traj$;
14:             else $\mathcal{T}.Trim(\boldsymbol{x}_{fail})$
```

A tree $\mathcal{T}$ is built from the initial state $\boldsymbol{x}_0$ by taking a random sample $\boldsymbol{x}_{rand} \in \mathbb{R}^3$ in the state-space ($RandomSample$). The dimensionality of $\boldsymbol{x}_{rand}$ is 3 because it involves the x-position and the incoming velocity. The parent node $\boldsymbol{x}_{parent}$ of the random sample $\boldsymbol{x}_{rand}$ is found ($FindParent$) based on a distance metric described in Section 5.3.2. A steer function ($STEER$) attempts to extend from $\boldsymbol{x}_{parent}$ to $\boldsymbol{x}_{rand}$, and a new node $\boldsymbol{x}_{new}$ will be added to the tree if no collision was detected during the $STEER$ step. Otherwise, a sample will be drawn. Details of the $STEER$ function is presented in Algorithm 2 and in Section 5.3.2. The kinodynamic RRT is terminated once the goal region has been reached ($ReachGoal$).

Then a path $path$ will be extracted from the tree ($\mathcal{T}.FindPath$).

The path is post-processed to eliminate redundant jumps to get $path^*$ ($path.Shortcut$) as described in Section 5.3.3. Trajectory optimization $TrajOpt$ is performed on the jumps in $path^*$ as described in Section 5.3.4. If it succeeds, we are done, but if this fails, the failed state is returned. In this case, the sub-tree rooted at the failure state will be deleted and the random sampling resumes. random sampling resumes.

5.3.2 Reachability-Informed Control Sampling

A distance metric $\rho(x_1, x_2)$ is used to find the nearest neighbor of the sample. We use a weighted Euclidean distance with weights $[10, 30, 0.1, 0.01]$ to account for the importance of horizontal and vertical components of position and velocity. The weight ratio of p_x and p_z affects how much the planner prefers to stay on the same height, and the weight on v_x is higher than v_z because v_x determines the direction of the jump and is sensitive to friction limits. The $STEER$ function finds a collision-free parabola that goes from $\boldsymbol{x}_{parent}$ towards $\boldsymbol{x}_{rand}$ while considering the velocity reachability map. First, an outgoing velocity $\boldsymbol{v}_{guess}$ is obtained by kinematically connecting $\boldsymbol{p}_{parent}$ and $\boldsymbol{p}_{rand}$ without velocity constraint ($Connect$). Second, $\boldsymbol{v}_{guess}$ is projected ($Project$) to the reachable set $\mathcal{R}(v_{parent})$, which is a convex polytope. Hence, the projection can be performed by solving a quadratic program (QP) [15].

The $STEER$ function tries at most N_{try} times to obtain a collision-free parabola. The random samples from $SampleWithBias$ follow a Gaussian distribution centered at $\boldsymbol{v}_{proj}$ with standard deviation of σ, where σ is the distance between $\boldsymbol{v}_{proj}$ and the farthest vertex of the polytope $\mathcal{R}(\boldsymbol{v}_{parent})$. Any sample outside of the reachable set is rejected. The sampled velocity $\boldsymbol{v}_{sample}$ is used to project $\boldsymbol{x}_{parent}$ to the landing state $\boldsymbol{x}_{new}$ ($Projectile$). $\boldsymbol{x}_{new}$ is returned if the parabola does not induce collision. Otherwise, $null$ is returned to indicate collision.

Algorithm 2 `STEER`

```
Input: $\mathcal{R}, \boldsymbol{x}_{rand}, \boldsymbol{x}_{parent}, Terrain$
Output: $\boldsymbol{x}_{new}$
1: $\boldsymbol{v}_{guess} \leftarrow Connect(\boldsymbol{p}_{rand}, \boldsymbol{p}_{parent})$
2: $\boldsymbol{v}_{proj} \leftarrow Project(\boldsymbol{v}_{guess}, \mathcal{R}(\boldsymbol{v}_{parent}))$
3: for $i = 1$ to $N_{try}$ do
4:     $\boldsymbol{v}_{sample} \leftarrow SampleWithBias(\boldsymbol{v}_{proj}, \mathcal{R}(\boldsymbol{v}_{parent}))$
5:     $collision, \boldsymbol{x}_{new} \leftarrow Projectile(\boldsymbol{x}_{parent}, \boldsymbol{v}_{sample}, Terrain)$
6:     if not $collision$ then
7:         return $\boldsymbol{x}_{new}$
8: return $null$
```

5.3.3 Path Shortcut

Once the goal region has been reached, a kinematic *path* is extracted from the tree $\mathcal{T}$ using the method $\mathcal{T}$.*FindPath*. Since it is possible that *path* involves redundant jumps, the *path.Shortcut* method is applied to shortcut the path and reduce the solve time for the subsequent TO. The *Shortcut* method is summarized in Algorithm 3. The node $\boldsymbol{x}_i$ attempts to connect with a node $\boldsymbol{x}_j (j > i)$ whose index starts from the end of *path*. If $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are successfully connected by the function $ConnectWith\mathcal{R}$, then the *Projectile* function is called to check for collision. If the parabola is collision free, then the new node $\boldsymbol{x}_{new}$ is added to the $path^*$ and $\boldsymbol{v}^i_{out}, \boldsymbol{v}^j_{in}$ will be updated. Then the index i is given the value of j to indicate a shortcut. In the outer loop, $\boldsymbol{x}_i$ marches from the start of *path* sequentially, until it reaches the length of the *path* given by the method *path.Length*. The function $ConnectWith\mathcal{R}$ solves an small nonlinear program that accounts for the velocity reachability map,

$$\begin{aligned}
\underset{\boldsymbol{v}^i_{out}, \boldsymbol{v}^j_{in}, T_{air}}{\text{minimize}} \quad & J && (5.2a)\\
\text{subject to} \quad & \boldsymbol{v}^i_{out} \in \mathcal{R}(\boldsymbol{v}^i_{in}) && (5.2b)\\
& \boldsymbol{v}^j_{in} \in \mathcal{R}^{-1}(\boldsymbol{v}^j_{out}) && (5.2c)\\
& \boldsymbol{x}^j_{in} = \boldsymbol{\Phi}(\boldsymbol{x}^i_{out}, T_{air}), && (5.2d)
\end{aligned}$$

where the objective function J is set to a constant to form a feasibility problem.

Algorithm 3 `Path Shortcut`

Input: $path, \mathcal{R}, Terrain$

Output: $path^*$

```
1: $path^*.init(path[0])$
2: $N \leftarrow path.Length(), i \leftarrow 0$
3: while $i < N$ do
4:     for $j = N$ to $i + 1$ do
5:         $\boldsymbol{x}_i \leftarrow path[i], \boldsymbol{x}_j \leftarrow path[j]$
6:         $success = False$
7:         $connected, \boldsymbol{v}_{out} \leftarrow ConnectWith\mathcal{R}(\boldsymbol{x}_i, \boldsymbol{x}_j, \mathcal{R})$
8:         if $connected$ then
9:             $collision, \boldsymbol{x}_{new} \leftarrow Projectile(\boldsymbol{x}_i, \boldsymbol{v}_{out}, Terrain)$
10:            if not $collision$ then
11:                $success = True$
12:        if $success = True$ then
13:            $path^*.add(\boldsymbol{x}_{new})$
14:            $i \leftarrow j$
15:            break
16:        else if $j = i + 1$ then
17:            $path^*.add(\boldsymbol{x}_j)$
18:            $i++$
19: return $path^*$
```

5.3.4 Trajectory Optimization

The parabola sequence $path^*$ from the sampling-based planner is solved in sequence using calls to TO^1 to connect each subsequent parabola with a stance phase. If TO^1 cannot solve a connection, a new TO that considers two consecutive jumps (TO^2) will be solved. This attempts to connect the incoming velocity from the current stance phase to the outgoing velocity in the next stance phase, which involves trajectory optimization over two stance phases and an aerial phase.

$$
\begin{aligned}
\underset{\boldsymbol{\alpha}_F^i, T_{st}^i, T_{air}, \boldsymbol{x}_{LO}^1, \boldsymbol{x}_{TD}^2}{\text{minimize}} \quad & \sum_{i=1}^{2}\sum_{k=1}^{N} ||\boldsymbol{\tau}_k^i|| \cdot T_{st}^i && \text{(5.3a)} \\
\text{subject to} \quad & \boldsymbol{p}_k^i \in \Omega && \text{(5.3b)} \\
& \tau_{min} \leq \boldsymbol{\tau}_k^i \leq \tau_{max} && \text{(5.3c)} \\
& \boldsymbol{x}_{TD}^1 \in \boldsymbol{P}_{in}^1, \boldsymbol{x}_{LO}^2 \in \boldsymbol{P}_{out}^2 && \text{(5.3d)} \\
& \boldsymbol{x}_{TD}^2 = \boldsymbol{\Phi}(\boldsymbol{x}_{LO}^1, T_{air}) && \text{(5.3e)} \\
& \boldsymbol{F}_k^i \in \mathcal{C}(\mu), && \text{(5.3f)}
\end{aligned}
$$

where the superscript $i \in \{1, 2\}$ indicates the stance sequence. The aerial phase with aerial time T_{air} that connects stance 1 and 2 has the kinematic relationship $\boldsymbol{\Phi}$,

$$
\boldsymbol{\Phi}(\boldsymbol{x}, T) = \begin{bmatrix} \boldsymbol{p} + \boldsymbol{v}T + \frac{1}{2}\boldsymbol{a}_g T^2 \\ \boldsymbol{v} + \boldsymbol{a}_g T \end{bmatrix}. \tag{5.4}
$$

TO^2 can sometimes find trajectories that cannot be discovered by TO^1 since TO^2 has more relaxed constraints. If TO^2 cannot find a solution, the current stance state is treated as causing the failure and will be returned as $\boldsymbol{x}_{fail}$ to be pruned from the tree.

5.4 Results

5.4.1 Computation Setup

The hybrid planning framework is formulated in MATLAB, and the TO^1 (5.1) and TO^2 (5.3) are formulated in CasADi [2] using the multiple-shooting method. The resulting nonlinear program (NLP) is solved by the solver ipopt [9]. The computational geometry calculation is done using the Mutli-Parametric Toolbox 3 (MPT3) [53]. The QPs for projection as described in Section 5.3.2 are solved by qpSWIFT [106]. The MICP-based planner in Section 5.4.3 is implemented using YALMIP [83] and solved by gurobi [102]. All of the simulation examples are run on a desktop with Intel i7 at 3.40 GHz.

5.4.2 Performance on Various Terrains

The proposed hybrid motion planner is tested on 5 different terrains, as shown in Fig. 5.4 and Fig. 5.5. The terrains are assumed to be represented by piecewise constant functions. The robot starts from the initial foot location (green circle) and tries to find a jumping path to reach the goal region (yellow box). Within each stance phase, the touchdown velocity $\boldsymbol{v}_{TD}$ is indicated by a blue arrow and the liftoff velocity $\boldsymbol{v}_{LO}$ a red arrow. The gray region shown in terrains (a) and (b) are the workspaces. The initial and final robot configurations at each stance are shown for terrains (a)-(d). Note that in terrain (d) segment 4, the planner adopted the strategy of taking intermediate jumps to re-orient the momentum of the robot in order to jump over the wide gap. Fig. 5.5 shows an example solution where the robot takes 9 jumps to reach the goal region. At stance 2, the TO^1 failed to find a solution, which is indicated by the black dot. Nevertheless, TO^2 found a feasible trajectory by simultaneously solving for stance 2 and 3 on segment 3. A zoom-in view is presented to illustrate stance 2 and 3, where the aerial trajectory is shown in a black dotted line. Fig. 5.6 (a) summarizes the solve time decomposition for the example terrains. Solve time results from 20 trials are averaged and the standard deviation is represented by the error bar. Terrain (a) takes the shortest time (5 s) and the terrain in Fig. 5.5 takes the longest time to solve (26 s). Fig. 5.6 (b) shows the average node number of the RRT tree. Please note that the y-axis is in log scale.

Although our method solves these problems in tens of seconds, the performance of the current implementation can be significantly improved. First, it is coded in MATLAB for rapid prototyping, so its run time can be reduced once re-written in a compile language. In addition, the run time can be further decreased if the TO at each stance was parallelized since each stance can be solved independently.

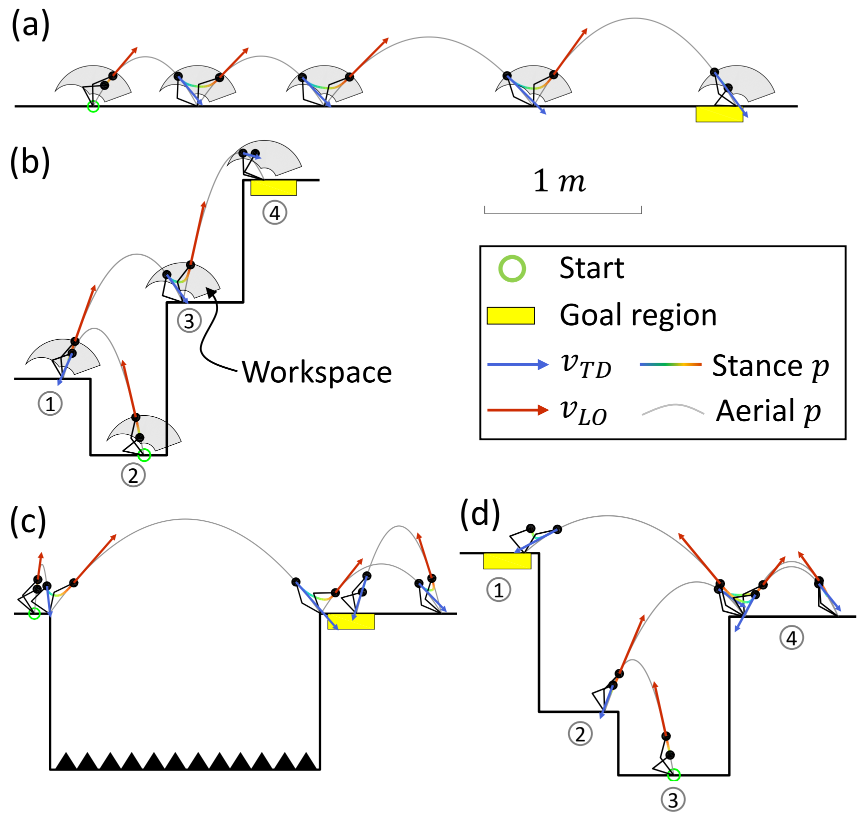


Figure 5.4: Example of terrains that are solved using the proposed hybrid motion planner. A scale is presented to show dimension. (a) a flat ground (b) three stairs: Left-Right-Right (c) a wide gap, where the region with saw teeth is forbidden (d) three stairs: Left-Right-Left.

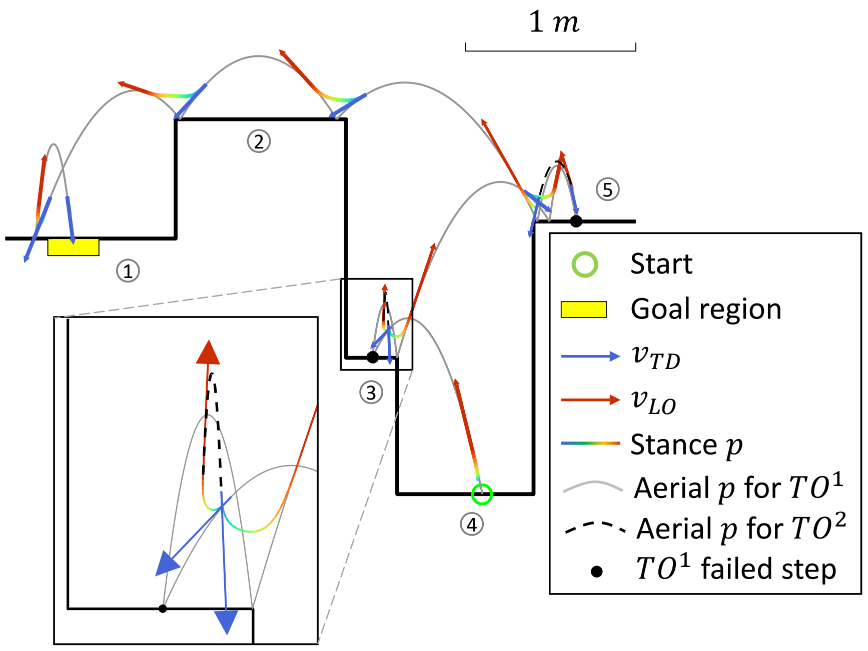


Figure 5.5: The hybrid sampling/optimization-based planning algorithm can solve complex terrains as the one presented in this figure. The zoom-in part shows the case where TO^1 failed at stance 2, and TO^2 succeeded by solving for stance 2 and 3 simultaneously.

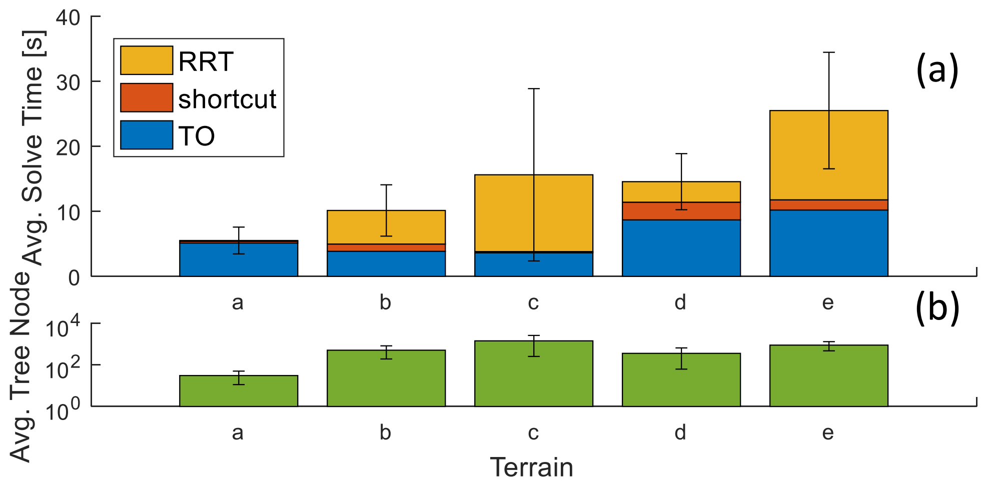


Figure 5.6: Simulation results from 20 trials on the example terrains. The error bar represents one standard deviation. (a) The average solve time decomposed into three parts, RRT, shortcut, and TO. (b) The average node number of the tree.

5.4.3 Benchmark

The proposed hybrid motion planning framework is compared with a quasi-static planner and a mixed-integer convex program (MICP) based planner. The three methods are tested in the scenario presented in Fig. 5.7 (a). The platform height $h, 2h$ and the gap width w are varied and the solve time results are shown in Fig. 5.7 (b)(c).

Quasi-static Planning

This method is similar to the proposed one except that it does not utilize the velocity reachability map introduced in Section 5.2.2 in the sampling stage. The quasi-static planning assumes that each step starts with zero velocity and can achieve maximum velocity v_{max} in every direction. The second assumption entails the assumed reachable region to be a parabolic envelope parametrized by v_{max} [39].

Mixed-Integer Convex Programs

The MICP-based method here plans consecutive jumps while considering actuator limits [31]. It is a resolution complete algorithm whose worst-case solve time increases drastically as the number of jumps increases. The number of jump is set to 2 for the MICP-based planner to limit the solve time within the same order of magnitude as the other two methods. To achieve convex formulation, the torque limit constraint is relaxed and the solution is more conservative.

Fig. 5.7 (b) presents the solve time of the three methods where w is fixed at 0.4 m and h is varied from 0.2 m to 0.7 m. As can be observed from the figure, the MICP-based planner is slower than the other two methods and failed when h is higher than 0.45 m. In contrast, the quasi-static planner performs almost as well as the hybrid planner in the first scenario. Fig. 5.7 (c) shows the solve time comparison where h is fixed at 0.5 m, and w is varied from 0.3 m to 0.6 m. The quasi-static planner failed at w =0.43 m due to its static assumption. The MICP-based planner can solve a slightly wider gap at the cost of longer solve time. In

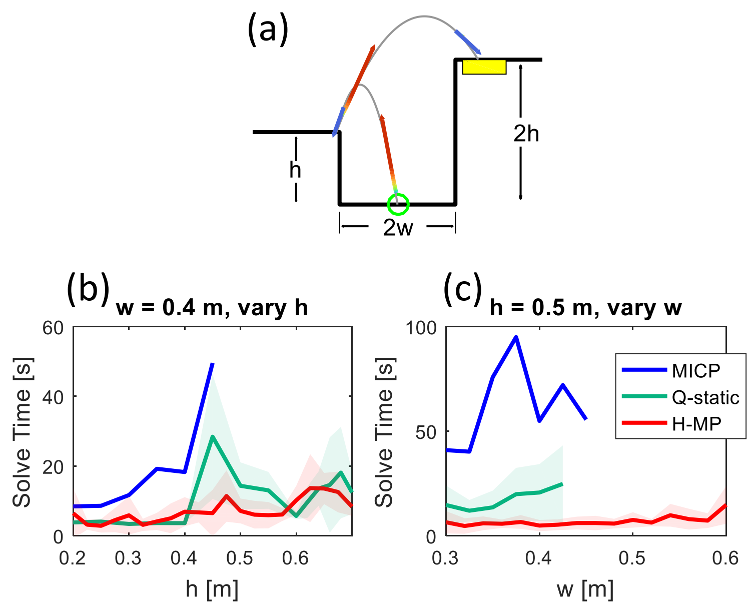


Figure 5.7: Benchmark result of the hybrid planner, quasi-static planner, and MICP-based planner. The former two are repeated 10 times and the average is shown as a solid line and the shaded area represents one standard deviation. (a) The scenario for the benchmarking of the three methods (b) The testing case where w is fixed at 0.4 m and h is varied (c) The testing case where h is fixed at 0.5 m, and w is varied.

comparison, the proposed hybrid planner finds solutions in all the tested scenarios using the shortest solve time.

These tests highlight the advantages of the proposed hybrid planning framework. Compared with the quasi-static planner, the proposed framework utilizes the velocity reachability map to reason about the momentum of the robot. Hence, the proposed method can come up with the strategy of taking intermediate jumps to re-direct its momentum to overcome wide gaps. The MICP-based planner can plan for consecutive jumps but its solve time does not scale well as the number of jump increases due to the curse of dimensionality. Besides, the MICP-based planner produces conservative results due to the convex relaxation.

5.4.4 Experiment Setup

The experimental platform used in this chapter is identical to the R1 robot used in Section 4.3.4 except that the total moving mass is 1.1 kg. The R1 robot is fixed to the end of a boom system with a radius 1.25 m. The position of the robot is measured by two encoders installed at the base of the boom. The feedforward force profile from the hybrid planning algorithm is applied at the stance phases, and a PD controller is applied during the aerial phase to track foot swing trajectory. Proprioceptive contact detection, as presented in Section 2.4.3, was implemented to initiate stance phases.

5.4.5 Experiment Result

The terrain is set up such that the robot has to make use of the 0.4 m high platform on the left to reach the goal region on the 0.9 m high platform on the right. The snapshots of the experiment are presented in Fig. 5.1. The hybrid planner can come up with the strategy of making intermediate jumps on the left platform to re-direct its momentum to clear the wide gap and reach the goal region.

The hip and knee joint torque trajectories for the three jumps experiment are shown in Fig. 5.8, where the stance phases are indicated by the gray areas. It can be observed that both hip and knee torques are within the torque limit (10 Nm). The oscillation after the stance phase is due to the rapid swing foot retraction to avoid collision with the environment.

5.5 Summary

This chapter presents a hybrid sampling/optimization motion planning algorithm for an agile single-legged robot to jump over challenging terrains. Under appropriate assumptions, the original kinodynamic motion planning problem could be decoupled into sampling and optimization stages. In the sampling stage, a variant of the kinodynamic RRT algorithm is employed to search for a kinematically feasible path as a sequence of parabolas. The pre-computed velocity reachability map is utilized to restrain the samples to be within a subset of the state space, which accelerates the algorithm by increasing the success rate of the subsequent TO. After a path shortcutting procedure, the optimization stage solves a

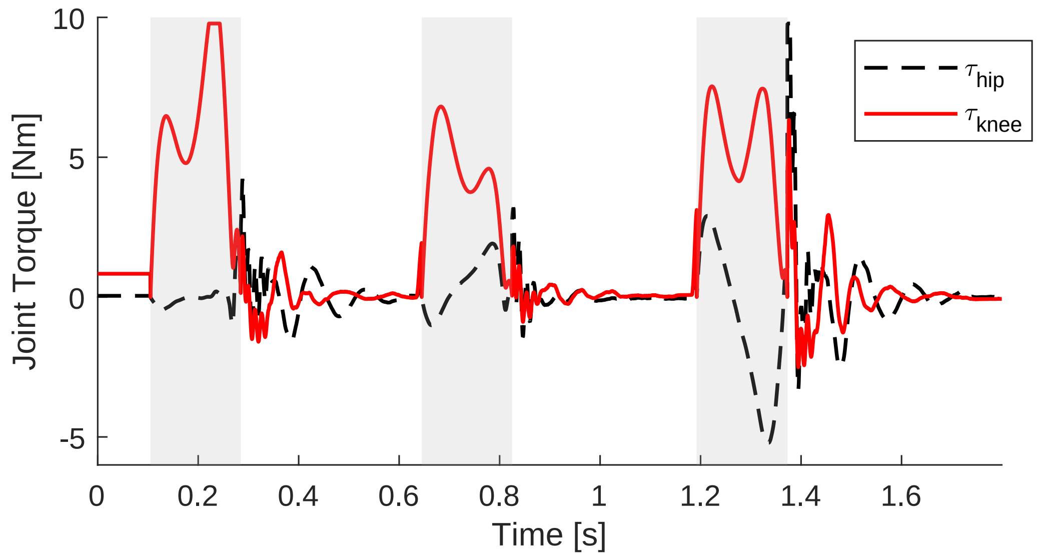


Figure 5.8: The joint torque recording for the three consecutive jumps experiment. The shaded areas indicate stance phases.

TO problem at each jump for the dynamically feasible trajectory. The performance of the proposed hybrid motion planning algorithm is shown on various example terrains, and the advantage of this method is highlighted through benchmarking with two other methods. A trajectory generated by the proposed method is applied on a physical robot, which successfully traversed a challenging terrain by executing 3 consecutive jumps. The proposed hybrid planner is applicable to other single-legged robots such as SALTO [46] to traverse more complex terrains.

Chapter 6

Summary and Conclusion

6.1 Summary

The major advantage of legged robots is the ability to navigate complex and unstructured environments. Inspired by the locomotion competence of agile animals, the goal of this thesis is to make contribution towards developing motion capabilities for legged robots, particularly for negotiating complex environments. This thesis has provided an integrative framework that encompasses control, motion planning, and hardware synthesis of legged robot by leveraging the optimization-based methods. In general, this thesis has made broad contributions in areas of simple-model-based control and planning for legged robots. Specifically, we proposed the representation-free model predictive control (RF-MPC) which can stabilize quadruped robot acrobatic motions in $SE(3)$ under a unified framework, and a kinodynamic motion planning framework that simultaneously plans contact and centroidal dynamics, while explicitly enforcing the actuator torque constraint. Throughout this thesis, hardware experiments are conducted on various custom legged robot platforms to validate the proposed methods. Chapter 2 details the synthesis process of a dynamic quadrupedal robot *Panther* with the custom high-power proprioceptive actuator. The QP-based controller and off-line trajectory optimization (TO) enabled the squat jumping experiment with a maximal jumping height of 0.7 m.

In Chapter 3, the quadrupedal robot *Panther* serves as the experimental platform for the RF-MPC. By directly using the rotation matrix, this framework opens up the possibility to stabilize complex 3D acrobatic maneuvers that may involve singularities in the widely-used

Euler angles representation. Experiment results of various gaits and a controlled tumble demonstrated that the RF-MPC controller can stabilize various dynamic motions that may involve singularity within a single control framework.

In Chapter 4, we formulate the mixed-integer convex program (MICP) to solve the kinodynamic motion planning problems for dynamic legged robot to traverse complex terrains. Structure of each individual problem is exploited while simple models are utilized to mitigate the problem of high computational requirement. We replace the non-convex constraints in the original TO problem with piece-wise convex relaxations to formulate a MICP. The MICP is then solved by off-the-shelf numerical solvers to obtain dynamic trajectories with global optimality certificate, given the discretization resolution. Experiment results on two robot platforms demonstrate the efficacy of the MICP framework for enabling aperiodic dynamic legged locomotion to overcome complex obstacles.

In Chapter 5, long-horizon motions are obtained by using a hybrid sample / optimization - based motion planning algorithm. This hybrid algorithm decouples the problem into a sampling and an optimization stage by employing a motion primitive. We use a variant of the kinodynamic RRT algorithm to search for kinematically-feasible path, where a pre-computed velocity reachability map is employed to accelerate the process. After a path short-cutting procedure, the optimization stage solves a TO problem at each jump to obtain a dynamically-feasible trajectory. Simulation and experiment results demonstrate that the robot successfully traversed a variety of challenging terrains by executing consecutive jumps.

6.2 Future Work

The progress made in this thesis can serve as a starting point for future robotics research. Here we list a few possible research directions:

- The RF-MPC framework is likely to open up possibilities for controlling extremely dynamic 3D motions for ground and aerial robots. With the emergence of powerful and light-weight computing units, the RF-MPC formulation can be applied to stabilizing acrobatic maneuvers in UAVs. For legged robots, we envision to equip them with special end-effectors, such as claws [110] or magnetic grippers, to enable them to climb up vertical surfaces and walk on ceilings.

- Although the MICP framework has been shown to be applicable for various challenging terrains, its long solve time prohibits it from being implemented in a receding horizon fashion. Recent work of Marcucci and Tedrake [84] proposes a method that warm-starts MIQP in hybrid MPC applications. Development of robust MIQP algorithms suitable for embedded applications [130] and integration with machine learning [5] may open up opportunities for real-time hybrid MPC in legged robots.
- The possible application of real-time MIQP solvers can unlock a legion of new dynamic and intelligent behaviors for legged locomotion. We envision a cascade control framework where a high-level kinodynamic motion planner generates long horizon behavior by solving MIQP at a lower frequency; A mid-level MPC operating at a medium frequency for refining motion plans in shorter behavior horizon, possibly by leveraging NLP solvers (e.g. SQP, DDP); A low-level QP-based controller for motion regulation at a high frequency.

6.3 Conclusion

This thesis has developed an integrative framework that tightly connects the control, planning and hardware synthesis of legged robots for negotiating challenging environments through aperiodic dynamic maneuvers, largely through the application of optimization-based methods. To track complex 3D acrobatic motions, we developed a representation-free MPC (RF-MPC) framework that can stabilize dynamic motions with large orientation excursion, in a unified framework. This approach is fundamentally different from the prevalent Euler angle representation since the direct utilization of the rotation matrix avoids the singularity issue of Euler angles, and does not require switching among controllers. This unified framework is beneficial because switching between controllers is either slow or prone to edge case failure. This thesis also makes contribution in legged robot motion planning algorithms. Prior to this work, state-of-the-art algorithms either solve the footstep planning and the dynamic trajectory tracking problems in a sequential manner or solve for dynamic trajectories and contact simultaneously in a single NLP. The first approach disregards the interplay between footstep placement and centroidal motion, and searches for a trajectory in a restricted subset of the solution space. The second approach relies on solving a computationally expensive NLP whose solution is sensitive to the initial guesses. Our MICP-based planner utilizes simple templates and considers contact, dynamics, and torque limit concurrently, while not requiring any initial guesses. To further push the behavior horizon, the hybrid sample/optimization-based planner leverages the advantages of both regimes and reduces the computational expense for overcoming complex terrains.

Looking ahead, parallel advancement of sensing technology, efficient numerical solvers, together with the framework proposed in this thesis have the potential to endow legged robot with autonomous mobility in complex unstructured environment for real-world deployment.

Bibliography

[1] 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. *IEEE Robotics and Automation Letters*, 3(3):2531–2538, July 2018.

[2] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[3] Neculai Andrei. A sqp algorithm for large-scale constrained optimization: Snopt. In *Continuous Nonlinear Optimization for Engineering Applications in GAMS Technology*, pages 317–330. Springer, 2017.

[4] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[5] Dimitris Bertsimas and Bartolomeo Stellato. Online mixed-integer optimization in milliseconds. *arXiv preprint arXiv:1907.02206*, 2019.

[6] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[7] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming.* SIAM, 2010.

[8] Sanjay P Bhat and Dennis S Bernstein. A topological obstruction to global asymptotic stabilization of rotational motion and the unwinding phenomenon. In *American Control Conference, 1998. Proceedings of the 1998*, volume 5, pages 2785–2789. IEEE, 1998.

[9] Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[10] Gerardo Bledt. *Regularized predictive control framework for robust dynamic legged locomotion.* PhD thesis, Massachusetts Institute of Technology, 2020.

[11] Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. MIT cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

[12] Gerardo Bledt, Patrick M Wensing, and Sangbae Kim. Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 4102–4109. IEEE, 2017.

[13] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[14] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 521–528. IEEE, 2000.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[16] Tim Bretl, Stephen Rock, Jean-Claude Latombe, Brett Kennedy, and Hrand Aghazarian. Free-climbing with a multi-use robot. In *Experimental Robotics IX*, pages 449–458. Springer, 2006.

[17] Francesco Bullo and Andrew D Lewis. *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*, volume 49. Springer Science & Business Media, 2004.

[18] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. K nitro: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

[19] Mylene Campana and Jean-Paul Laumond. Ballistic motion planning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1410–1416. IEEE, 2016.

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

[21] Peter Corke. An inertial and visual sensing system for a small autonomous helicopter. *Journal of robotic systems*, 21(2):43–51, 2004.

[22] John J Craig. *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India, 2009.

[23] Xingye Da and Jessy Grizzle. Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots. *The International Journal of Robotics Research*, 38(9):1063–1097, 2019.

[24] Stefano Dafarra, Sylvain Bertrand, Robert J Griffin, Giorgio Metta, Daniele Pucci, and Jerry Pratt. Non-linear trajectory optimization for large step-ups: Application to the humanoid robot atlas. pages 3884–3891, 2020.

[25] Hongkai Dai, Gregory Izatt, and Russ Tedrake. Global inverse kinematics via mixed-integer convex optimization. *The International Journal of Robotics Research*, 38(12-13):1420–1441, 2019.

[26] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with simple dynamics and full kinematics. In *Proceedings of the IEEE-RAS international conference on humanoid robots*, 2014.

[27] Alessandro De Luca and Raffaella Mattone. Sensorless robot collision detection and hybrid force/motion control. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 999–1004. IEEE, 2005.

[28] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 279–286, Nov 2014.

[29] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[30] Yanran Ding, Chuanzheng Li, and Hae-Won Park. Kinodynamic motion planning for multi-legged robot jumping via mixed-integer convex program. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[31] Yanran Ding, Chuanzheng Li, and Hae-Won Park. Single leg dynamic motion planning with mixed-integer convex optimization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6. IEEE, 2018.

[32] Yanran Ding, Abhishek Pandala, Chuanzheng Li, Young-Ha Shin, and Hae-Won Park. Representation-free model predictive control for dynamic motions in quadrupeds. *IEEE Transactions on Robotics*, pages 1–18, 2021.

[33] Yanran Ding, Abhishek Pandala, and Hae-Won Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8484–8490. IEEE, 2019.

[34] Yanran Ding and Hae-Won Park. Design and experimental implementation of a quasi-direct-drive leg for optimized jumping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 300–305. IEEE, 2017.

[35] Yanran Ding, Mengchao Zhang, Chuanzheng Li, Hae-Won Park, and Kris Hauser. Hybrid sampling/optimization-based planning for agile jumping robots on challenging terrains. In *2021 International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[36] Çetin Dişibüyük and Halil Oruç. A generalization of rational bernstein–bézier curves. *BIT Numerical Mathematics*, 47(2):313–323, 2007.

[37] Eid H. Doha, Ali H. Bhrawy, and M. A. Saker. Integrals of Bernstein polynomials: An application for the solution of high even-order differential equations. *Applied Mathematics Letters*, 24(4):559–565, 2011.

[38] Bruce Donald, Patrick Xavier, John Canny, and John Reif. Kinodynamic motion planning. *Journal of the ACM (JACM)*, 40(5):1048–1066, 1993.

[39] Denis Donnelly. The parabolic envelope of constant initial speed trajectories. *AmJPh*, 60(12):1149–1150, 1992.

[40] Roy Featherstone. *Rigid body dynamics algorithms.* Springer, 2014.

[41] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. Optimization-based full body control for the darpa robotics challenge. *Journal of Field Robotics*, 32(2):293–312, 2015.

[42] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

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

[44] Yukai Gong, Ross Hartley, Xingye Da, Ayonga Hereid, Omar Harib, Jiunn-Kai Huang, and Jessy Grizzle. Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway. In *2019 American Control Conference (ACC)*, pages 4559–4566. IEEE, 2019.

[45] Alexander Graham. *Kronecker products and matrix calculus with applications.* Courier Dover Publications, 2018.

[46] Duncan W Haldane, Mark M Plecnik, Justin K Yim, and Ronald S Fearing. Robotic vertical jumping agility via series-elastic power modulation. *Science Robotics*, 1(1), 2016.

[47] Duncan W Haldane, Justin K Yim, and Ronald S Fearing. Repetitive extreme-acceleration (14-g) spatial jumping with salto-1p. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3345–3351. IEEE, 2017.

[48] Kris Hauser, Timothy Bretl, Kensuke Harada, and Jean-Claude Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Algorithmic foundation of robotics VII*, pages 507–522. Springer, 2008.

[49] Kris Hauser, Timothy Bretl, and J-C Latombe. Non-gaited humanoid locomotion planning. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 7–12. IEEE, 2005.

[50] Kris Hauser, Timothy Bretl, Jean-Claude Latombe, Kensuke Harada, and Brian Wilcox. Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research*, 27(11-12):1325–1349, 2008.

[51] Bernd Henze, Christian Ott, and Maximo A Roa. Posture and balance control for humanoid robots in multi-contact scenarios based on model predictive control. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3253–3258. IEEE, 2014.

[52] Martin Herceg, Michal Kvasnica, Colin N. Jones, and Manfred Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013.

[53] Martin Herceg, Michal Kvasnica, Colin N Jones, and Manfred Morari. Multi-parametric toolbox 3.0. In *2013 European Control Conference (ECC)*, pages 502–510. IEEE, 2013.

[54] Andrei Herdt, Holger Diedam, Pierre-Brice Wieber, Dimitar Dimitrov, Katja Mombaur, and Moritz Diehl. Online walking motion generation with automatic footstep placement. *Advanced Robotics*, 24(5-6):719–737, 2010.

[55] Andrei Herdt, Nicolas Perrin, and Pierre-Brice Wieber. Walking without thinking about it. In *IROS 2010-IEEE-RSJ International Conference on Intelligent Robots & Systems*, pages 190–195. IEEE, 2010.

[56] Ayonga Hereid, Eric A Cousineau, Christian M Hubicki, and Aaron D Ames. 3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1447–1454. IEEE, 2016.

[57] Ayonga Hereid, Christian M Hubicki, Eric A Cousineau, Jonathan W Hurst, and Aaron D Ames. Hybrid zero dynamics based multiple shooting optimization with applications to robotic walking. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5734–5740. IEEE, 2015.

[58] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016.

[59] Alexander Herzog, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Trajectory generation for multi-contact momentum control. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 874–880. IEEE, 2015.

[60] 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 *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44. IEEE, 2016.

[61] Marco Hutter, Hannes Sommer, Christian Gehring, Mark Hoepflinger, Michael Bloesch, and Roland Siegwart. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research*, 33(8):1047–1062, 2014.

[62] Jemin Hwangbo, Vassilios Tsounis, Hendrik Kolvenbach, and Marco Hutter. Cable-driven actuation for highly dynamic robotic systems. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8543–8550. IEEE, 2018.

[63] IBM Corp. User's manual for CPLEX, 2010.

[64] Matt R Jardin and Eric R Mueller. Optimized measurements of unmanned-air-vehicle mass moment of inertia with a bifilar pendulum. *Journal of Aircraft*, 46(3):763–775, 2009.

[65] S Kajita. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In *Proc. IEEE Int. Conf. on Robotics and Automation, Sacramento, CA, 1991*, pages 1405–1411, 1991.

[66] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[67] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[68] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[69] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[70] Gavin Kenneally, Avik De, and Daniel E Koditschek. Design principles for a family of direct-drive legged robots. *IEEE Robotics and Automation Letters*, 1(2):900–907, 2016.

[71] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[72] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.

[73] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 692–698. IEEE, 2001.

[74] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Motion planning for humanoid robots. In *Robotics Research. The Eleventh International Symposium*, pages 365–374. Springer, 2005.

[75] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40(3):429–455, 2016.

[76] Boris Lau, Christoph Sprunk, and Wolfram Burgard. Kinodynamic motion planning for mobile robots using splines. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2427–2433. IEEE, 2009.

[77] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. *Report No. TR 98-11, Computer Science Department, Iowa State University.*, 1998.

[78] Steven M LaValle. *Planning Algorithms*. Cambridge university press, 2006.

[79] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[80] Taeyoung Lee, Melvin Leoky, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on SE(3). In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5420–5425. IEEE, 2010.

[81] Chuanzheng Li, Yanran Ding, and Hae-Won Park. Centroidal-momentum-based trajectory generation for legged locomotion. *Mechatronics*, 68:102364, 2020.

[82] Min Liu, Zherong Pan, Kai Xu, and Dinesh Manocha. New formulation of mixed-integer conic programming for globally optimal grasp planning. *IEEE Robotics and Automation Letters*, 5(3):4663–4670, 2020.

[83] Johan Lofberg. YALMIP: A toolbox for modeling and optimization in matlab. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

[84] Tobia Marcucci and Russ Tedrake. Warm start of mixed-integer programs for model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 2020.

[85] Jerrold E Marsden and Tudor S Ratiu. Introduction to mechanics and symmetry. *Physics Today*, 48(12):65, 1995.

[86] 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 *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1096–1103. IEEE, 2017.

[87] Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G Caldwell, and Claudio Semini. Hierarchical planning of dynamic movements without scheduled contact sequences. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4636–4641. IEEE, 2016.

[88] Maxon Motor. Maxon motor catalog. [Online] Available: www.maxonmotorusa.com/.

[89] Christopher G. Mayhew, Ricardo G. Sanfelice, and Andrew R. Teel. On quaternion-based attitude control and the unwinding phenomenon. In *Proceedings of the 2011 American Control Conference*, pages 299–304, 2011.

[90] Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part I-convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

[91] Edward S Meadows, Michael A Henson, John W Eaton, and James B Rawlings. Receding horizon control and discontinuous state feedback stabilization. *International Journal of Control*, 62(5):1217–1229, 1995.

[92] Michael Mistry, Jonas Buchli, and Stefan Schaal. Inverse dynamics control of floating base systems using orthogonal decomposition. In *2010 IEEE international conference on robotics and automation*, pages 3406–3412. IEEE, 2010.

[93] Michael Mistry and Ludovic Righetti. Operational space control of constrained and underactuated systems. *Robotics: Science and systems VII*, pages 225–232, 2012.

[94] MIT-Biomimetics-Robotics-Lab. Cheetah-software. https://github.com/charlespwd/project-title[Accessed 29 June 2020], 2019.

[95] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

[96] James Morgan. How far can a squirrel jump? https://birdwatchingbuzz.com/how-far-can-a-squirrel-jump/[Accessed 23 Feb. 2021].

[97] MOSEK ApS. The MOSEK optimization software, 2014.

[98] DM Murray and SJ Yakowitz. Differential dynamic programming and newton's method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, 43(3):395–414, 1984.

[99] Michael Neunert, Farbod Farshidian, Alexander W Winkler, and Jonas Buchli. Trajectory optimization through contacts and automatic gait discovery for quadrupeds. *IEEE Robotics and Automation Letters*, 2(3):1502–1509, 2017.

[100] 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. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.

[101] Quan Nguyen, Matthew J Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Optimized jumping on the MIT Cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454, May 2019.

[102] Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[103] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous robots*, 35(2-3):161–176, 2013.

[104] Romeo Orsolino, Michele Focchi, Carlos Mastalli, Hongkai Dai, Darwin G Caldwell, and Claudio Semini. Application of wrench-based feasibility analysis to the online trajectory optimization of legged robots. *IEEE Robotics and Automation Letters*, 3(4):3363–3370, 2018.

[105] SS Osder, WE Rouse, and LS Young. Navigation, guidance, and control systems for V/STOL aircraft. *Sperry Tech*, 1(3), 1973.

[106] Abhishek Goud Pandala, Yanran Ding, and Hae-Won Park. qpswift: A real-time sparse quadratic program solver for robotic applications. *IEEE Robotics and Automation Letters*, 4(4):3355–3362, 2019.

[107] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. High-speed bounding with the MIT Cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36(2):167–192, 2017.

[108] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. Jumping over obstacles with MIT cheetah 2. *Robotics and Autonomous Systems*, 136:103703, 2021.

[109] Hae-Won Park, Patrick M Wensing, Sangbae Kim, et al. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. *Robotics: Science and Systems*, 2015.

[110] Jaejun Park, Hae-Won Park, et al. Design of anti-skid foot with passive slip detection mechanism for conditional utilization of heterogeneous foot pads. *IEEE Robotics and Automation Letters*, 4(2):1170–1177, 2019.

[111] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[112] Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1366–1373. IEEE, 2016.

[113] Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 200–207. IEEE, 2006.

[114] Joao Ramos and Sangbae Kim. Humanoid dynamic synchronization through whole-body bilateral feedback teleoperation. *IEEE Transactions on Robotics*, 34(4):953–965, 2018.

[115] Ludovic Righetti, Jonas Buchli, Michael Mistry, Mrinal Kalakrishnan, and Stefan Schaal. Optimal distribution of contact forces with inverse-dynamics control. *The International Journal of Robotics Research*, 32(3):280–298, 2013.

[116] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Inverse dynamics control of floating-base robots with external constraints: A unified view. In *2011 IEEE international conference on robotics and automation*, pages 1085–1090. IEEE, 2011.

[117] Srikanth Saripalli, Jonathan M Roberts, Peter Corke, Gregg Buskey, and Gaurav Sukhatme. A tale of two helicopters. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003 (IROS 2003)*, volume 1, pages 805–810. IEEE, 2003.

[118] Brian W Satzinger, Chelsea Lau, Marten Byl, and Katie Byl. Tractable locomotion planning for robosimian. *The International Journal of Robotics Research*, 34(13):1541–1558, 2015.

[119] Tom Schouwenaars, Bart De Moor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path planning. In *Control Conference (ECC), 2001 European*, pages 2603–2608. IEEE, 2001.

[120] Claudio Semini, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of HyQ–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.

[121] Luis Sentis. *Synthesis and control of whole-body behaviors in humanoid systems.* Citeseer, 2007.

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

[123] Luis Sentis, Jaeheung Park, and Oussama Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Transactions on robotics*, 26(3):483–501, 2010.

[124] Sangok Seok, Albert Wang, Meng Yee Chuah, David Otten, Jeffrey Lang, and Sangbae Kim. Design principles for highly efficient quadrupeds and implementation on the MIT cheetah robot. In *2013 IEEE International Conference on Robotics and Automation*, pages 3307–3312. IEEE, 2013.

[125] Sangok Seok, Albert Wang, David Otten, and Sangbae Kim. Actuator design for high force proprioceptive control in fast legged locomotion. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1970–1975. IEEE, 2012.

[126] Alexander Shkolnik, Michael Levashov, Ian R Manchester, and Russ Tedrake. Bounding on rough terrain with the littledog robot. *The International Journal of Robotics Research*, 30(2):192–215, 2011.

[127] Alexander Shkolnik, Matthew Walter, and Russ Tedrake. Reachability-guided sampling for planning under differential constraints. In *2009 IEEE International Conference on Robotics and Automation*, pages 2859–2865. IEEE, 2009.

[128] Malcolm D Shuster. A survey of attitude representations. *Navigation*, 8(9):439–517, 1993.

[129] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics.* Springer, 2016.

[130] Bartolomeo Stellato, Vihangkumar V Naik, Alberto Bemporad, Paul Goulart, and Stephen Boyd. Embedded mixed-integer quadratic optimization using the osqp solver. In *2018 European Control Conference (ECC)*, pages 1536–1541. IEEE, 2018.

[131] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[132] Francisco Trespalacios and Ignacio E Grossmann. Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie Ingenieur Technik*, 86(7):991–1012, 2014.

[133] Jeffrey C Trinkle, J-S Pang, Sandra Sudarsky, and Grace Lo. On dynamic multi-rigid-body contact problems with coulomb friction. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 77(4):267–279, 1997.

[134] Andrés Klee Valenzuela. *Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain.* PhD thesis, Massachusetts Institute of Technology, 2016.

[135] Gokul Varadhan and Dinesh Manocha. Accurate Minkowski sum approximation of polyhedral models. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pages 392–401. IEEE, 2004.

[136] Oskar von Stryk. *Numerical Solution of Optimal Control Problems by Direct Collocation*, pages 129–143. Birkhäuser Basel, Basel, 1993.

[137] Oskar Von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of operations research*, 37(1):357–373, 1992.

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

[139] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, March 2010.

[140] Patrick M Wensing and David E Orin. Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *2013 IEEE International Conference on Robotics and Automation*, pages 3103–3109. IEEE, 2013.

[141] Patrick M Wensing and David E Orin. High-speed humanoid running through control with a 3d-slip model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5134–5140. IEEE, 2013.

[142] Patrick M Wensing, Albert Wang, Sangok Seok, David Otten, Jeffrey Lang, and Sangbae Kim. Proprioceptive actuator design in the MIT Cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *IEEE Transactions on Robotics*, 33(3):509–522, 2017.

[143] Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

[144] Andrew Witkin and Michael Kass. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4):159–168, 1988.

[145] Philip Wolfe. The simplex method for quadratic programming. *Econometrica: Journal of the Econometric Society*, pages 382–398, 1959.

[146] Guofan Wu and Koushil Sreenath. Variation-based linearization of nonlinear systems evolving on SO(3)and S2,. *IEEE Access*, 3:1592–1604, 2015.

[147] Zhaoming Xie, Glen Berseth, Patrick Clary, Jonathan Hurst, and Michiel van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246. IEEE, 2018.

[148] Xiaobin Xiong and Aaron Ames. Dynamic and versatile humanoid walking via embedding 3d actuated slip model with hybrid lip based stepping. *IEEE Robotics and Automation Letters*, 5(4):6286–6293, 2020.

[149] Xiaobin Xiong and Aaron Ames. Slip walking over rough terrain via h-lip stepping and backstepping-barrier function inspired quadratic program. *IEEE Robotics and Automation Letters*, 6(2):2122–2129, 2021.

[150] Xiaobin Xiong and Aaron D Ames. Bipedal hopping: Reduced-order model embedding via optimization-based control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3821–3828. IEEE, 2018.

[151] Xiaobin Xiong and Aaron D Ames. Orbit characterization, stabilization and composition on 3D underactuated bipedal walking via hybrid passive linear inverted pendulum model. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4644–4651. IEEE, 2019.

[152] Xiufeng Yang, Ying Chen, Longlong Chang, Ariel A Calderón, and Néstor O Pérez-Arancibia. Bee+: A 95-mg four-winged insect-scale flying robot driven by twinned unimorph actuators. *IEEE Robotics and Automation Letters*, 4(4):4270–4277, 2019.

[153] Justin K Yim and Ronald S Fearing. Precision jumping limits from flight-phase control in salto-1p. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2229–2236. IEEE, 2018.

[154] Justin K Yim, Bajwa Roodra Pratap Singh, Eric K Wang, Roy Featherstone, and Ronald S Fearing. Precision robotic leaping and landing using stance-phase balance. *IEEE Robotics and Automation Letters*, 5(2):3422–3429, 2020.

Appendix A

Integration of Bézier Polynomial

A Bézier polynomial is a linear combination of a Bernstein polynomial basis [36], so the integration of a Bézier polynomial is a linear operation [37] on the Bézier coefficients. For example, the linear relationship between wrench Bézier coefficients and twist Bézier coefficients is

$$\frac{M+1}{T_{st}}\boldsymbol{\Phi}(M,T_{st})\boldsymbol{\alpha}_{\dot{q}} = [\boldsymbol{D}^{-1}\boldsymbol{\alpha}_{\mathcal{F}}^{\top} + \boldsymbol{a}_g, \dot{\boldsymbol{q}}_0]^{\top}, \tag{A.1}$$

where M is the order of Bézier polynomial; T_{st} is stance duration; $\dot{\boldsymbol{q}}_0 \in \mathbb{R}^3$ is the initial body twist; $\boldsymbol{\alpha}_{\dot{q}} \in \mathbb{R}^{(M+2)\times 3}$ is the Bézier coefficients for the spatial twist trajectory; $\boldsymbol{\Phi} \in \mathbb{R}^{(M+2)\times(M+2)}$ is a matrix whose elements are defined as

$$\boldsymbol{\Phi}_{i,j} := \begin{cases} -1, & j = i = 1, 2, \cdots, M+1 \\ 1, & j = i+1 = 2, 3, \cdots, M+2 \\ \frac{T_{st}}{M+1}, & i = M+2, j = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{A.2}$$

The linear operation $\boldsymbol{\alpha}_{\dot{q}} = \mathcal{L}(\boldsymbol{\alpha}_{\mathcal{F}}, \dot{\boldsymbol{q}}_0)$ is obtained by inverting the matrix in front of $\boldsymbol{\alpha}_{\dot{q}}$ in (A.1). Similarly, the Bézier coefficients of the configuration trajectory $\boldsymbol{q}(t)$ can also be integrated given initial configuration $\boldsymbol{q}_0 \in \mathbb{R}^3$.

Appendix B

Gain values for RF-MPC

The gain values for the simulation and experiment of RF-MPC are presented in Table B.1 and Table B.2, respectively. From experience, we found out that the gain matrices Q and R also have a range of values that lead to stable behaviors. However, the exact effect of each gain value is difficult to quantify due to a large number of tuning parameters and their complex interaction in objective landscape shaping. Nevertheless, we found some general guidelines or "rules of thumb" concerning the tuning of the gain matrices Q and R.

1. The Q_p, Q_R terms are comparable to the proportional terms in a PD controller.
2. The $Q_{\dot{p}}, Q_\omega$ terms are comparable to the derivative terms in a PD controller.
3. Increasing R_u makes the control follow the reference control more closely.
4. The order of the gain value normalizes the magnitude of the nominal error in each dimension to the same order. For example, $Q_p : (0.01m)^2 \cdot 1e5 = 10, Q_{\dot{p}} : (1ms^{-1})^2 \cdot 10 = 10, R_u : (10N)^2 \cdot 0.1 = 10$.

In practice, we start the gain value from a reasonable guess based on the guidelines, and then tune the gain values in a trial-and-error manner. From experience, we found out that an elaborate fine-tuning process is required only for complex maneuvers; simple motions such as trotting only require a relatively small amount of tuning effort.

Table B.1: Cost function weights for the **simulations**. The values in parenthesis represent weights on the terminal costs.

	Sim. Pose	Sim. TrotWalk	Sim. Bound	Acro. Mnvr.
Q_{p_x}	3e5 (1e5)	1e5	8e4	5e6
Q_{p_y}	5e5 (1e5)	2e5	5e4	5e6
Q_{p_z}	2e5 (1e5)	3e5	3e6	5e6
$Q_{\dot{p}_x}$	10 (30)	5e2	4e3(5e2)	5e3
$Q_{\dot{p}_y}$	8 (30)	1e3	5e2	5e3
$Q_{\dot{p}_z}$	10 (30)	1e3	7e2(5e2)	5e3
Q_{R_x}	5e2	1e3	8e3	1e6
Q_{R_y}	2e3 (3e3)	1e4	5e5 (5e4)	1e6
Q_{R_z}	1e3	8e2	8e3	1e6
Q_{ω_x}	2	40	2e2	5e3
Q_{ω_y}	4	40	1e2	5e3
Q_{ω_z}	3	10	2e2	5e3
R_{u_x}	0.1	0.1	0.2	0.1
R_{u_y}	0.1	0.2	0.2	0.1
R_{u_z}	0.1	0.1	0.2	0.1
T_{st}	N/A	0.3	0.1	0.1 (0.2)
T_{sw}	N/A	0.15	0.16	N/A
N_{hor}	7	6	6	7
γ	1.0	1.0	0.9	0.9
T_{pred}	0.05	0.08	0.01	0.01
f_{MPC}	100	100	100	100

Note: T_{st}, T_{sw} and T_{pred} all have the unit of [s]; N_{hor} is the MPC prediction horizon; T_{pred} is the prediction time step; f_{MPC} is the MPC control frequency with the unit of [Hz].

Table B.2: Cost function weights for the **experiments**. The values in parenthesis represent weights on the terminal costs.

	Exp. Pose/ Balance	Exp. TrotWalk	Exp. TrotRun	Exp. Bound	Exp. Backflip
Q_{p_x}	3e5 (1e5)	1e5	1e5	2e5 (1.2e5)	1e5
Q_{p_y}	5e5 (1e5)	1e5 (1.5e5)	1e5 (1.5e5)	4e5	1e5 (2e5)
Q_{p_z}	2e5 (1e5)	1.5e5 (2.2e5)	2e4	1.5e5 (2e5)	1.5e5 (2.2e5)
$Q_{\dot{p}_x}$	10 (30)	1e3 (1.5e3)	1e3 (1.5e3)	50	1e3 (1.5e3)
$Q_{\dot{p}_y}$	8 (30)	1e3	1e3	200 (150)	1e3
$Q_{\dot{p}_z}$	10 (30)	150	100	30	150
Q_{R_x}	5e2	2e3	1e3 (2e3)	3e3 (1e3)	4e3 (6e3)
Q_{R_y}	2e3 (3e3)	2e3	2e3	4e3 (8e3)	0 (10)
Q_{R_z}	1e3	8e2	8e2	1e3 (3e3)	8e2
Q_{ω_x}	2	60 (100)	60 (100)	3 (2)	60 (100)
Q_{ω_y}	4	40 (45)	40 (45)	6 (2)	0 (1)
Q_{ω_z}	3	10	10	5 (8)	10
R_{u_x}	0.1	0.1	0.1	0.1	0.1
R_{u_y}	0.1	0.18	0.18	0.18	0.12
R_{u_z}	0.1	0.08	0.08	0.2	0.1
T_{st}	N/A	0.3	0.12 / 0.2	0.1	0.12 (0.3)*
T_{sw}	N/A	0.15	0.2 / 0.1	0.2	0.3
N_{hor}	6	6	6	7	6
γ	1.0	1.0	1.0	0.8	0.8
T_{pred}	0.02	0.08	0.05	0.01	0.02
f_{MPC}	250	250	250	160	200

Note: T_{st}, T_{sw} and T_{pred} all have the unit of [s]; N_{hor} is the MPC prediction horizon; T_{pred} is the prediction time step; f_{MPC} is the MPC control frequency with the unit of [Hz].
*0.13 s is the front stance time, and 0.3 s is the hind stance time.