CONTROLLING CONTACT TRANSITIONS FOR DYNAMIC ROBOTS

William Yang

A DISSERTATION

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2024

Supervisor of Dissertation

Michael Posa, Assistant Professor of Mechanical Engineering and Applied Mechanics

Graduate Group Chairperson

Prashant K. Purohit, Professor of Mechanical Engineering and Applied Mechanics

Dissertation Committee

Nadia Figueroa, Assistant Professor of Mechanical Engineering and Applied Mechanics
Michael Posa, Assistant Professor of Mechanical Engineering and Applied Mechanics
Daniel E. Koditschek, Professor of Electrical and Systems Engineering
Aaron Johnson, Associate Professor of Mechanical Engineering, Carnegie Mellon University

ACKNOWLEDGEMENT

I would like to thank my advisor, Michael Posa, who taught me how to do true scientific research, to inquire one level deeper and always ask why. With endless patience, Michael gave me the space to understand things deeply and not gloss over even the smallest details. The kindness and wisdom in the mentorship I received from Michael is something I strive to replicate in my future career.

I am extremely grateful to my labmates at the DAIRlab. Mathew Halm, for helping establish the relaxed lab culture that persists today, but always insisting on the highest standards. Yu-Ming Chen, who took on the monumental task of starting the Cassie controller stack. Alp Aydinoglu, for always having time to chat. My closest collaborator, Brian Acosta, for his immense contributions to our shared codebase, dairlib, and for our frequent discussions, which saved countless headaches. Bibit Bianchini, whose quick thinking and intuition were a great asset in brainstorming. Finally to Leon Kim and Hien Bui, for asking tough questions. I am also grateful to the members of Kodlab, Shane Rozen-Levy, Diego Caporale, and Tim Greco for insightful cross lab discussions on legged locomotion. I would also like to thank the staff of the machine shop, especially Peter Szczesniak, for all their hardware assistance with Cassie and the Franka Panda.

Endless thanks to Yevgeniy Yesilevskiy and C. David Remy for welcoming me and mentoring me despite my inexperience. My immensely positive experience from my time in the RAMlab contributed greatly to my decision to pursue and PhD and join the greater robotics research community.

Thank you to my committee members for their expert experience and advice, which helped shape this thesis. Special thanks to Dan Koditschek, who always pushed me to view things from new perspectives.

I would like to thank my family for their endless love and support, and my sister Angela for pushing me to go after my dreams. The most special thanks to Jessica - for our thoughtful discussions and being by my side, both personally and professionally.

ABSTRACT

CONTROLLING CONTACT TRANSITIONS FOR DYNAMIC ROBOTS

William Yang

Michael Posa

Legged robots, robotic manipulators, and their combined embodiment as humanoid robots have received considerable attention across both academia and industry. However, with few notable exceptions, state-of-the-art demonstrations are significantly less dynamic than their biological counterparts. A formidable challenge towards achieving more dynamic robots lies within controlling contact interactions with their environment. Legged robots undergoing impacts experience near-instantaneous changes in their velocities, making accurate state estimation difficult and resulting in controller sensitivity to even small deviations in impact timing. Contact transitions are also challenging for robot manipulation due to the combinatorial complexity of planning across multiple contact modes. Frictional contact that often arises from dynamic manipulation further increases this planning complexity due to the introduction of additional contact modes and increased degree of underactuation.

To address these limitations, this thesis proposes algorithmic and systems contributions to gracefully handle contact transitions for dynamic robots. First, we identify that uncertainties from impact events enter the system dynamics in a structured manner. We leverage this structure to propose a general modification to model-based feedback controllers, enabling selective robustness to impact uncertainty while maximally retaining control authority. We validate our approach on custom dynamic jumping and running controllers on the 3D bipedal robot, Cassie. Then, we examine dexterous dynamic manipulation through complex non-prehensile tasks that require considering the full spectrum of hybrid contact modes. We leverage recent advancements in contact-implicit MPC to generate contact-rich motion plans in real-time. We demonstrate, through careful integration of the MPC and low-level tracking controller, how contact-implicit MPC can be adapted to dynamic tasks. We perform two distinct tasks using the same model, notably without common aids such as

reference trajectories or motion primitives, highlighting the generality of our approach.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1

Introduction

Legged robots, robotic manipulators, and their combined embodiment as humanoid robots have received considerable attention across both academia and industry. These robots have the promise of performing valuable tasks such as search and rescue as well as alleviating anticipated labor shortages in caretaker and logistics fields. However, current robotic applications are siloed to tightly controlled environments in industrial manufacturing settings. With few exceptions, state-of-the-art demonstrations are stiff and showcase considerably less agility than their biological counterparts. As robots transition from research projects to integral components of our lives, we want to develop methods that can employ robots to their full physical capabilities. By working towards methods that can reason about and regulate the robot's complete set of dynamics, we expand the locomotion and manipulation skills to include those that can only be accomplished dynamically, such as running and jumping.

While decades of control theory have developed rich and capable feedback controllers for *continuous* systems, robots must interact with their environment through contact. These contacts introduce forces and constraints that produce distinct contact modes, each with different governing dynamics, which combined form a *hybrid* system. The governing dynamics within a single contact mode are continuous, so prior theory can be applied. However, controlling transitions across these contact modes is comparatively underexplored.

Controlling for these contact transitions is challenging because the governing dynamics change instantly across contact modes, and thus the optimal action for one contact mode may be suboptimal in another. This is further complicated when attempting to control for dynamic motions. At impact events, velocities exhibit near discontinuities (Fazeli et al., 2020), which make accurate state measurements difficult. For this reason, controllers are particularly sensitive to uncertainty in timing during these impact events (Saccon et al., 2014). Due to the time scale of these impact dynamics, common strategies such as contact detection (Bledt et al., 2018b) are insufficient.

Leveraging the robot's full dynamics also introduces sliding or frictional contact modes. Frictional contact is challenging due to increased combinatorial complexity and model inaccuracies. Reasoning about hybrid decisions such as contact and no-contact is already challenging due to the combinatorial complexity that originates from consideration of multiple actions and therefore contact modes in the future (Cheng et al., 2022) (Graesdal et al., 2024). The addition of sliding as a possible contact mode further increases the complexity of this analysis. The boundary between sliding and sticking contact is challenging to regulate. While the boundary between contact and no-contact is purely configuration dependent, the separation of sliding and sticking contact is more complex. Controlling for frictional contact is made even more challenging because there are known inaccuracies in our models of frictional contact (Chatterjee, 1997) (Remy, 2017) (Halm and Posa, 2019). These inaccuracies necessitate a reactive approach, which introduces real-time computational constraints. The combinatorial complexity of planning through the full spectrum of contact modes combined with the computational constraints, motivates simplifications in the form of offline computation such as motion primitives (Woodruff and Lynch, 2017) or reference trajectories (Le Cleac'h et al., 2024); however, this limits the generalization capabilities of these methods.

In this thesis, we embrace the complexity of impact dynamics and frictional contact. By understanding the structure of uncertainty in impacts, we develop a principled method to make our controllers robust while maximally retaining control authority. Reconciling with the inaccuracies of frictional contact models, we leverage and augment contact-implicit MPC to react to and re-plan contact-rich motions in real-time.

1.1. Outline and Contributions

In Chapter 2, we give an overview of related work for controlling dynamic systems with contact. We review the current state-of-the-art capabilities in dynamic bipedal locomotion, contact-rich manipulation, and dynamic manipulation. We also review relevant algorithms in robustness to contact uncertainty as well as planning and control of contact-rich and dynamic manipulation.

In Chapter 3, we provide the mathematical background used to model robot systems with contact and the formulations used to control these systems.

The primary contributions of this thesis begin in Chapter 4. Here, we focus on the challenge of applying low-level feedback control to discontinuous tracking objectives resulting from impact forces. To address this, we introduce impact-invariant control, a general method for adapting feedback controllers to be robust to uncertainty present at impact events. We demonstrate our method on the first model-based box-jumping and running controllers formulated for Cassie.

In Chapter 5, we shift our focus to manipulation. We propose a dynamic manipulation task that requires the exploration of the full spectrum of hybrid contact modes. This task not only includes frequent making and breaking contact, but also the under-utilized transitions between sticking and sliding contact. We leverage recent breakthroughs in contact-implicit MPC to automatically reason about the contact modes, and carefully integrate the MPC with the downstream tracking controller to accurately track the dynamic plans including forceful object interactions. Finally, in Chapter 6, we finish with final discussions about directions for future work.

CHAPTER 2

Related Work

In this chapter, we review related work in dynamic bipedal locomotion, methods for handling contact uncertainty, and contact-rich manipulation.

2.1. Dynamic Bipedal Locomotion

In this section, we review the current state-of-the-art in dynamic 3D bipedal locomotion. Note, while impressive dynamic capabilities have also been demonstrated on planar bipeds (Hodgins and Raibert, 1988; Sreenath et al., 2012; Ma et al., 2017), these robots lack complexity which simplifies control. For example, these robots use near massless point feet, which leads to more manageable impact dynamics. Impressive feats have also been demonstrated on robots such as Asimo (Hirose and Ogawa, 2007) and Atlas (Dynamics, 2021). However, the methods that enable these behaviors are poorly documented because they are industrial research projects. For this reason, the majority of recent publications on dynamic 3D bipedal locomotion were developed for Cassie. Cassie, a compliant 3D bipedal robot designed and manufactured by Agility Robotics[1], is equipped with an actuated blade foot as well as yaw degrees of freedom and thus capable of balancing in place and turning. Reinforcement learning (RL) controllers developed for Cassie showcased the full-spectrum of bipedal gaits (Siekmann et al., 2021a) (Li et al., 2024) with impressive dynamicism and robustness. Although these methods can generate impressive capabilities, the mechanisms responsible for creating the behavior are still unclear. For example, it was hypothesized that increased foot clearance and a swing-leg retraction aided in enabling blind traversal of stairs (Siekmann et al., 2021b), but the policy also demonstrated complex recovery maneuvers that are not captured in this justification. Thus any pitfalls and limitations of the RL policies are not easily understood without extensive evaluation on the actual robot. In contrast, the assumptions and therefore limitations of model-based controllers can be explicitly defined. For example, controllers that plan with Cassie's full dynamics by pre-computing gait libraries (Reher and Ames, 2021) (Gong et al., 2019) are limited by the poor scaling of needing to compute a separate trajectory for each operator command. To

[1] https://agilityrobotics.com/

achieve planning rates fast enough for real-time control, methods leverage reduced order models at the cost of reduced expressivity (Chen and Posa, 2020). However, the limitations of the reduced order models can be addressed. As an example, Gong and Grizzle (2022) motivated replacing the Linear Inverted Pendulum (LIP) template with a more predictive model, Angular Momentum Linear Inverted Pendulum (ALIP). Gibson et al. (2022) extended ALIP from flat terrain to sloped terrain and Dosunmu-Ogunbi et al. (2023) made further modifications to support stairs. Despite these efforts, model-based controllers still have not achieved Cassie's full dynamic capabilities, evidenced by the gap from the RL policies, indicating further exploration is still required.

2.2. Robustness To Contact Uncertainty

Although legged robots must be robust to many sources of uncertainty such as model uncertainty in parameters like mass, inertia, and friction as well as external force perturbations from external loads, legged robots are particularly sensitive to contact uncertainty. While model uncertainty and force perturbations influence just the continuous dynamics, contact uncertainty alters the hybrid dynamics. Because footstep placement is the primary mode of regulating the bulk motion of the robot, changes in the contact location or timing has significant effects on the overall motion.

Efforts to improve robustness to contact uncertainty largely fall into three complementary strategies: designing inherently robust open-loop trajectories, faster and more accurate contact detection, and addressing low-level controller sensitivity to discontinuous tracking objectives.

Regulating to trajectories that are inherently robust to contact uncertainty is a good place to start. Similar to how guinea fowls handle large unseen disturbances in ground height (Daley and Biewener, 2006), the open-loop swing-leg retraction trajectory coupled with spring-like leg dynamics was shown to have inherent stability to variable terrain heights (Seyfarth et al., 2003). In search of this property, multiple approaches have also rediscovered similar swing-leg retraction policies through explicitly optimizing for it using optimization methods (Dai and Tedrake, 2012) (Green et al., 2020) (Zhu et al., 2022). Additionally, through stochastic methods, roboticists have discovered methods for how to compute foot clearance through optimization (Drnach and Zhao, 2021).

Detecting off-nominal contact switches is important as the feedback strategies can differ greatly according to the current contact mode because the dynamics differs. For this reason, accurate and fast contact detection is important for reacting to uncertain contacts. Examples of improvements in contact detection include (Bledt et al., 2018b).

Despite robust trajectories and improved contact detection, controllers are still sensitive to contact transitions. This is due to the unavoidable differences between the reference trajectory and actual system state from compliance in true contacts and even minuscule differences in impact timing. These differences may cause feedback error to spike and constraints to be violated, leading to instability. Methods for avoiding these controller spikes can be heuristic, such as reducing or blending controller gains around the impact event (Mason et al., 2016) (Atkeson et al., 2015), but these do not directly address the sources of the discontinuities. A suite of strategies, called reference spreading control (Saccon et al., 2014) (Van Steen et al., 2024), addresses the issue of possibly invalid time-based reference trajectories for mismatches in impact timing. Additional extensions leverage dynamical systems-like vector fields to eliminate the time-dependency (van Steen et al., 2023). However, all of these methods still require contact detection or accurate velocity estimates (van Steen et al., 2023), which are both difficult to achieve during the impact event. For this reason, these methods include an intermediate mode during contact where velocity feedback is turned off.

Another impact-aware formulation is (Wang et al., 2023), which incorporates anticipated impacts when considering the constraints in the task-space controller. When evaluating velocity-dependent constraints near anticipated impact events, the impact-friendly controller considers both pre- and post- impact velocities as predicted by their impact model (Wang et al., 2022). By doing so, they avoid constraint infeasibilities that are purely due to mismatches in impact timing.

2.3. Contact-Rich Manipulation

In Chapter 5, we demonstrate a contact-rich dynamic manipulation task using contact-implicit MPC. While pick-and-place using rigid grasps significantly simplifies manipulation by eliminating the need for object state estimation once a grasp is achieved, it only represents a small fraction of the manipulation skills employed by humans. Contact-rich manipulation refers to the class of

manipulation skills where contact is an integral component. In this section we discuss relevant literature for contact-rich manipulation, beginning with the simplest non-prehensile manipulation skills and slowly adding complexity. We also organize this section by first discussing methods that utilize tailored models and controllers and finally finish by discussing contact-implicit methods, where the contact mode sequence is not specified a priori, enabling improved generalization to new tasks.

2.3.1. Non-Prehensile Manipulation

Non-prehensile manipulation expands the capabilities of robot manipulators by removing the requirement of establishing a rigid grasp. A motivating example is the waiter task, where one or multiple objects are balanced on top of a tray and transported without requiring the objects to be rigidly fixed. Many renditions of this task have been performed. Subburaman et al. (2023) included a residual tilt angle at the end effector to reduce sliding. Brei et al. (2024) leveraged reachability analysis to generate provable motions where the object will not slide despite model uncertainty. (Heins and Schoellig, 2023) included strict non-sliding constraints to MPC to reactively plan trajectories while keeping the objects upright. However, only considering sticking contact is limiting, as sliding contact is an essential component to basic manipulation skills such as pushing.

Sliding or frictional contact is notoriously difficult to model. Even in simple systems such as those in planar pushing (Mason, 1986), dedicated methods are used to predict frictional interactions (Bauza et al., 2018). Also for planar systems, Chavan-Dafle et al. (2020) demonstrated in-grasp repositioning using external contacts via motion cones, and simultaneous estimation Doshi et al. (2022) (Taylor et al., 2023) of the boundary between sticking and sliding contact has been demonstrated as well.

While quasi-static models are sufficient for many pushing and pivoting tasks, sliding contact is also relevant to dynamic manipulation (Mason and Lynch, 1993). For example, Shi et al. (2017) demonstrated in-hand sliding for repositioning, regulating both inertial and contact forces. Similarly, Hou et al. (2020) demonstrated in-hand pivoting, robustly switching between two control strategies for sliding and sticking.

2.3.2. Contact-Implicit Methods

The methods introduced to this point leveraged models and controllers tailored to the specific task being performed, with an explicit control strategy for each contact mode. However, this approach is not scalable to the vast variety of manipulation tasks. For this reason, roboticists have turned to contact-implicit (Posa et al., 2014) methods for a single control framework that can reason across contact modes.

Although it enables a diverse set of manipulation tasks to be accomplished using the same framework (Cheng et al., 2022, 2023) or planning with global perspective (Graesdal et al., 2024), the additional complexity conflicts with real-time requirements. However, recent advancements have resulting in multiple contact-implicit methods fast enough for reactive control. These methods leverage diverse strategies ranging from offline gradients about a reference trajectory (Le Cleac'h et al., 2024), local gradients about a sampled trajectory (Howell et al., 2022a) (Kurtz et al., 2023), or iterative steps with non-physical constraints (Aydinoglu et al., 2024). The differences in the strategies and evaluation examples result in limited understanding of the fundamental limitations of the methods.

CHAPTER 3

Background

In this chapter, we provide the mathematical background for the control of legged robots and robot manipulators; commonly used symbols are listed in Table 3.1. First, in Section 3.1, we discuss how we model our robots as rigid body systems. We give an overview of the manipulator equations and how contacts impose constraints on the dynamics. Next, in Section 3.2, we discuss how the continuous dynamics for different contact modes are connected across contact transitions through rigid impacts. Section 3.3 details the hierarchical control strategy we use throughout this thesis. Finally, in Section 3.4 we discuss how we utilize simulation for controller design.

q	system configuration
v	system velocity
x	system state
u	system input
λ_c	contact force
λ_h	constraint force
$J_\lambda(q)$	contact Jacobian
$J_h(q)$	constraint Jacobian
$M(q)$	mass/inertia matrix
x^-	pre-impact state
x^+	post-impact state
$\mathcal{R}(q)$	impact reset map
Λ	contact/constraint impulse
y	task-space position
$\psi(q)$	task-space kinematics function
$\phi(q)$	signed distance between bodies
n_q	number of configuration variables
n_v	number of velocity variables
n_x	number of state variables
n_u	number of input variables
n_c	number of contact forces
n_h	number of constraint forces
μ	friction coefficient

Table 3.1: Commonly used symbols for rigid body dynamics with contacts.

3.1. Modeling of Rigid Body Systems

Legged robots, robot arms, and objects studied in this thesis can be well represented as trees of rigid linkages. The configuration q, or position of all parts of the robot, can be minimally described by the displacement of the joints between the linkages. In addition, the robots and objects are generally not constrained in how they move in their environment, and thus the configuration also includes floating-base degrees of freedom. To use the 3D compliant bipedal robot, Cassie, as an example, its configuration is composed of the floating-base position of its pelvis $q_{fb} = [q_{qw}, q_{qx}, q_{qy}, q_{qz}, q_x, q_y, q_z]$[2] expressed in the world frame along with the joint positions $q_{leg_L} = [q_{roll_L}, q_{yaw_L}, q_{pitch_L}, q_{knee_L}, q_{ankle_L}, q_{toe_L}]$, where we use the subscript $()_L$ and $()_R$ to denote the left and right legs respectively. The full configuration of Cassie is expressed as $q = [q_{fb}, q_{leg_L}, q_{leg_R}]$.

As another example, the configuration of the system containing a robot manipulator and object explored in Chapter 5, can be fully described as $q = [q_{arm}, q_{object}]$, where q_{arm} contains only the joint angles of the arm as its base is rigidly anchored to the world and $q_{object} = q_{fb}$ contains only its floating-base degrees of freedom because it has no articulation degrees of freedom.

3.1.1. System State

In this thesis, we use the minimal set of coordinates to describe the state of our robotic system. The state $x = [q, v]$ is comprised of the system configuration variables $q \in \mathbb{R}^{n_q}$ and the system velocities $v \in \mathbb{R}^{n_v}$. The floating-base velocities are expressed as $v_{fb} = [q_{wx}, q_{wy}, q_{wz}, q_{vx}, q_{vy}, q_{vz}]$, where q_{wx}, q_{wy}, q_{wz} are the angular velocities about the world x, y, and z axes respectively and q_{vx}, q_{vy}, q_{vz} are the translational velocities also expressed in the world frame.

Because we use quaternions when representing 3D orientations, the time derivatives of the positions, $\dot{q}$, are related to the velocities, v, by the map $N(q) \in \mathbb{R}^{n_q \times n_v}$,

$$\dot{q} = N(q)v. \tag{3.1}$$

[2]We use the following notation for quaternions. q_{qw} represents the real component of the quaternion and q_{qx}, q_{qy}, q_{qz} are the imaginary components of the quaternion, all expressed in the world frame. q_x, q_y, q_z is the position of the floating base also expressed in the world frame.

3.1.2. Continuous Dynamics: The Manipulator Equation

The dynamics of rigid body systems can be derived through the Euler-Lagrange equation and expressed in the following form known as the manipulator equation

$$M(q)\dot{v} + C(q, v) = Bu + J_\lambda(q)^T \lambda, \tag{3.2}$$

where $M(q) \in \mathbb{R}^{n_v \times n_v}$ is the mass matrix, $C(q, v) \in \mathbb{R}^{n_v}$ represents the Coriolis, centrifugal forces and gravitational forces, B is the selection matrix that maps actuation inputs $u \in n_u$ to generalized forces, and $J_\lambda(q) \in \mathbb{R}^{n_c \times n_v}$ is the Jacobian that maps external contact forces to generalized forces.

3.1.3. Contact Constraints for Continuous Dynamics

In the case of legged robots, we frequently wish to describe motions for when the contacting foot does not move relative to the environment, which is described as a constraint, specifically the no-slip constraint.

As implied by the name, this restricts or constrains the possible space of dynamics of the robot. Mathematically, we can express the position of the foot p_{foot} using a purely kinematic function $p_{foot} = \phi(q)$. Differentiating with respect to time, $\dot{p}_{foot} = \frac{\partial \phi}{\partial q}\frac{dq}{dt}$, we arrive at $\dot{p}_{foot} = J_\lambda v$, where $J_\lambda = \frac{\partial \phi}{\partial q}$.

Mathematically, this can be represented through a more minimal set of coordinates: those without the constrained degrees of freedom. However, for flexibility, in this thesis we choose to leave the general manipulator dynamics unchanged and instead include the no-slip constraint as a constraint on the system acceleration as

$$J_\lambda \dot{v} + \dot{J}_\lambda v = 0, \tag{3.3}$$

which is derived from differentiating $\dot{p}_{foot}$ with respect to time and constraining it to 0.

3.2. Rigid Contacts and Impacts

We model the complex deformations and surface forces that occur when a robot makes contact with other objects or the environment using a rigid-body contact model. This contact model does not allow deformations; instead, impacts are resolved instantaneously. Therefore, the configuration remains constant over the impact event, all forces except for the contact forces are considered negligible, and the velocities change instantaneously according to the contact impulse Λ:

$$M(v^+ - v^-) = J_\lambda^T \Lambda, \tag{3.4}$$

where v^- and v^+ are the pre- and post-impact velocities respectively, J_λ is the Jacobian for the active constraints of the new contact mode, and Λ is the impulse sustained over the impact event. If we include the no-slip constraint that the new stance foot does not move once in contact with the ground (purely inelastic collision and no-slip condition),

$$J_\lambda v^+ = 0, \tag{3.5}$$

Λ can be solved for explicitly, determining the post-impact state x^+ purely as a function of the pre-impact state x^-:

$$\begin{bmatrix} I & -M^{-1}J_\lambda^T \\ J_\lambda & 0 \end{bmatrix} \begin{bmatrix} v^+ \\ \Lambda \end{bmatrix} = \begin{bmatrix} v^- \\ 0 \end{bmatrix}, \tag{3.6}$$

which can be simplified to

$$q^+ = q^-, \tag{3.7}$$

$$v^+ = (I - M^{-1}J_\lambda^T(J_\lambda M^{-1}J_\lambda^T)^{-1}J_\lambda)v^-. \tag{3.8}$$

For conciseness, we define the mapping from pre-impact states to pose-impacts states using the state reset map $\mathcal{R}$:

$$x^+ = \mathcal{R}(x^-). \tag{3.9}$$

This reset map is commonly enforced as a constraint between hybrid modes separated by an impact event when modeling the hybrid dynamics of legged robots (Westervelt et al., 2003) (Reher and Ames, 2021) (Li and Wensing, 2020).

3.3. Control Hierarchy: High Level and Low Level Control

For the complex robotic systems explored in this thesis, abstractions are utilized to tractably control these systems. A common abstraction is separating the control problem into two parts: high-level planning over a larger set of decisions, and low-level local instantaneous feedback to stabilize the high-level plans.

3.3.1. Trajectory Optimization, MPC, and Templates

High-level planners are tasked with reasoning over a more global or broader perspective of the system, often requiring reasoning over a larger set of decision variables. This increased scope can take on many forms. For example, we may consider a longer perspective in time, reasoning about not only how to choose the current action, but also how to choose the next sequence of actions, taking into account future states. Another increase in scope could be consideration of other objects and obstacles in the environment.

Computing motion plans by reasoning over future actions is a common problem in robotics. While motions exist as continuous trajectories in time, we typically parameterize the trajectories as a discrete set of decision variables by discretizing in time, including the costs and constraints. This

enables us to formulate a general optimization problem:

$$\min_{z} \quad J(x_N) + \sum_{k=0}^{N} J(x_k, u_k, h) \tag{3.10}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k, h) \tag{3.11}$$

$$g(x_k, u_k) \geq 0, \tag{3.12}$$

where $z = [x_0, u_0, x_1, u_1, ..., x_N, u_N, h]$ is the vector of decision variables, $x_{k+1} = f(x_k, u_k, h)$ is the dynamics constraint imposed as each break point k, also commonly referred to as a knot point, and h is the timestep in between knot points. $g(x_k, u_k)$ represent additional constraints on the motion such as kinematic or actuator limits or restrictions used to define specific motions.

Note, the above formulation only considers a single contact mode. Formulating trajectory optimization for hybrid systems across multiple contact modes can be constructed by combining multiple trajectory optimization problems, each with its own set of dynamic constraints.

$$\min_{z^0,...,z^M} \quad \sum_{j=0}^{M} J(z^j) \tag{3.13}$$

$$\text{s.t.} \quad x^j_{k+1} = f^j(x_k, u_k, h) \tag{3.14}$$

$$g^j(x_k, u_k) \geq 0 \tag{3.15}$$

$$x^j_0 = \mathcal{R}(x^{j-1}_N). \tag{3.16}$$

We impose the rigid reset map (3.9) to ensure consistency between the optimization problems across the contact mode transitions. Variations of this problem include introducing additional variables to improve accuracy (Posa et al., 2016) or augmenting the problem to optimize for robustness (Drnach and Zhao, 2021) (Dai and Tedrake, 2012) (Green et al., 2020). Given the added complexity, these hybrid trajectory optimization problems cannot be solved quickly and typically require high quality initial guesses.

Beyond generating offline plans, trajectory optimization can also be used to synthesize online, also known as real-time, controllers and typically referred to as model predictive control (MPC). In order to make these problems tractable to solve online, the problem is simplified by utilizing reduced order models (Chen and Posa, 2020) (Bledt and Kim, 2019) (Chignoli et al., 2022) and often leverage linearization in order to transcribe it as a quadratic program (QP), a class of convex optimization problems with dedicated fast solvers.

Both trajectory optimization and MPC formulations require dynamics constraints (3.11) that are defined specific to a single contact mode, which requires that the contact mode sequence is specified a priori. This requirement of pre-specifying the contact mode sequence can be limiting when reasoning about systems with many potential contacts and knowing the optimal contact sequence is not obvious. Thus, allowing the contact sequence to be directly optimized over is a desirable property. Trajectory optimization formulations where the contact mode is implicitly defined by the dynamics constraints, and therefore does not need to be specified, is termed contact-implicit optimization (Posa et al., 2014). Formulations with this property are typically even more difficult to solve; however, recent work have generated formulation that are tractable to solve as real-time MPC (Kurtz et al., 2023) (Le Cleac'h et al., 2024) (Aydinoglu et al., 2024) (Shirai et al., 2024).

An alternative to optimal control is to utilize carefully chosen reduced order models or more formally known as templates (Full and Koditschek, 1999). These reduced order models aim to capture the essential underlying dynamics and are amenable to rigorous numerical or analytical stability analysis (Koditschek and Buehler, 1991) (Holmes et al., 2006). Canonical examples for legged robots include the linear inverted pendulum (LIP) (Kajita et al., 2001) or spring-loaded inverted pendulum (SLIP) (Poulakakis and Grizzle, 2009), with well studied regulation strategies (Raibert et al., 1984) (Seyfarth et al., 2003).

3.3.2. Operational Space Control

High-level plans often cannot be computed quickly enough to stabilize the systems. For this reason, we utilize low-level controllers that compute actions at high frequencies to stabilize and track the plans sent by the high-level planners. In contrast with high-level planners, low-level controllers

only reason about the current timestep. The objective of low-level controllers is to compute the instantaneous actuation forces u, typically in the form of motor torques, that best achieve the desired accelerations or forces. We use an operational space controller (OSC) as the low-level tracking controller for both Cassie and the Franka Panda. An OSC is a model-based inverse dynamics controller that tracks a set of task space accelerations by solving for dynamically consistent control inputs, contact forces, and generalized accelerations (Wensing and Orin, 2013) (Sentis and Khatib, 2005). The formulation using task space outputs is general, meaning joint-space inverse dynamics controllers (Gong and Grizzle, 2022) (Reher and Ames, 2021) and impedance control (Hogan, 1985) can be expressed in this formulation. We define an output y as a kinematic function of the robot configurations, meaning $y = \phi(q)$. We can then compute time derivative of the output by differentiating with respect to time as $\dot{y} = J_y(q)v$, where $J_y(q) = \frac{\partial \phi}{\partial q}$. The objective is to track a desired output reference trajectory y, which is achieved through proportional and derivative feedback in the output space. Specifically, we compute the commanded output acceleration $\ddot{y}_{cmd}$, which is calculated from the feedforward reference accelerations $\ddot{y}_{des}$ with proportional and derivative feedback:

$$\ddot{y}_{cmd} = \ddot{y}_{des} + K_p(y_{des} - y) + K_d(\dot{y}_{des} - \dot{y}), \tag{3.17}$$

where K_p and K_d are feedback gains on the output space position and velocity error respectively. The computed commanded output acceleration is not guaranteed to be dynamically feasible to achieve, so the objective of the OSC is then to minimize the difference between the achievable instantaneous output accelerations $\ddot{y}$ given by:

$$\ddot{y} = \dot{J}_y v + J_y \dot{v},$$

and the commanded output acceleration $\ddot{y}_{cmd}$.

This objective can be formulated as a quadratic program (QP):

$$\min_{u,\lambda,\dot{v}} \quad \sum_{i}^{N} \left\|\tilde{\ddot{y}}_i\right\|_{W_i} + \|u\|^2_{W_u} + \|\dot{v}\|^2_{W_{acc}} + \left\|\tilde{\lambda}\right\|^2_{W_\lambda} \tag{3.18}$$

$$\text{s.t.} \quad M\dot{v} + C = g + Bu + J_\lambda^T \lambda \tag{3.19}$$

$$J_h\dot{v} + \dot{J}_h v = 0 \tag{3.20}$$

$$J_\lambda\dot{v} + \dot{J}_\lambda v = 0 \tag{3.21}$$

$$|\lambda_x| \leq \mu\lambda_z \tag{3.22}$$

$$|\lambda_y| \leq \mu\lambda_z \tag{3.23}$$

$$\lambda_n \geq 0, \tag{3.24}$$

$$u_{min} \leq u \leq u_{max} \tag{3.25}$$

where i denotes the particular output being tracked (e.g., center of mass or foot position) and W_i are corresponding weights on tracking that output. The QP seeks to minimize the acceleration tracking error, $\tilde{\ddot{y}}_i = \ddot{y}_{i_{cmd}} - \ddot{y}_i$ and optionally the feedforward force, $\tilde{\lambda} = \lambda - \lambda_{des}$, for the weighted sum across all tracking objectives. Additional regularization costs can be added to avoid non-unique solutions if the problem is underspecified. (3.19) is the dynamics constraint, where J_λ is the Jacobian for the active contact constraints for the current mode, and λ are the corresponding constraint forces. (3.20) enforces the satisfaction of holonomic constraints such as the four-bar linkage on Cassie, where J_h is the Jacobian for the holonomic constraints. Similarly, (3.21) enforces that the points in contact cannot move. Additional constraints on the points in contact include that the contact forces remain in the friction cone (3.22), (3.23), and (3.24). Here, λ_z is the normal component of the contact force and λ_x, λ_y are the tangential components expressed in the robot frame. W_u, W_{acc}, and W_λ are general regularization weights on the decision variables, specifically the inputs, generalized accelerations, and contact forces respectively.

Note, this is the most general formulation that we consider, and some costs and constraints may be omitted depending on the relevant dynamics of the robot. For example, we do not consider friction

cone constraints, nor are there any holonomic constraints for the Franka Panda. For Cassie, we do not give references for the force variables; i.e. $\lambda_{des} = 0$. The QP can be solved quickly (>1000 Hz) on modern CPUs for both Cassie and the Franka Panda.

3.3.3. Contact Modes Transitions

Section 3.3.1 details methods for generating motion plans across a sequence of contact modes and Section 3.3.2 is a low-level tracking controller with the capability of tracking those motion plans for a specified contact mode. When the contact mode changes, the constraints Eqs. (3.19) to (3.21), (3.23) and (3.24) and tracking objectives (3.18) must be updated according to the active contact mode. For example, the friction cone constraints are only imposed for tracking Cassie's center of mass may be reasonable for a balancing controller, but there is no control authority over the center of mass when Cassie is in flight.

An intuitive solution would be to use the contact mode timings chosen by the motion planner. However, due to error from force perturbations, environment uncertainty, or simply tracking error the impact timing on the real system can deviate from the planned timings. In these situations, the active tracking objectives and constraints may be invalid and thus the low-level tracking controller may command suboptimal control actions that destabilize the system. Thus accurately fast contact mode detection (Bledt et al., 2018b) combined with reactive planning (Jeon et al., 2022) is important for mitigating these situations.

3.4. Simulation for Controller Design

Running experiments on the physical robot hardware is costly in terms of time as well as wear on physical components, which is particularly likely given the dynamic motions explored in this thesis. Beyond just reducing risk of damaging the robot, simulators can be a useful tool for principled controller design. Simulators are useful because they can be configured to start from any initial condition and provide ground-truth state information. Consistent initial conditions enable repeatable and scalable evaluation, which crucially provides consistent signals when tuning controller parameters. Ground-truth state information is useful because it eliminates uncertainty. This section details how we configure simulators to provide relevant signals for controller development and how

we design our controller architecture for seamless sim-to-real deployment. Note, this section does not discuss the use of simulation as part of a controller (Howell et al., 2022a; Kurtz et al., 2023).

3.4.1. Simulator Selection

There are many popular robotics simulators (Coumans, 2015; Makoviychuk et al., 2021; Todorov et al., 2012; Tedrake and the Drake Development Team, 2019). When choosing between simulators, important considerations include whether the simulator can simulate the necessary components for the particular application and the fidelity and performance of that simulation.

At their core, all robotics simulators are capable of simulating smooth rigid body dynamics through the implementation of the standard rigid body dynamics algorithms (Featherstone, 2014). Beyond integration accuracy, where differences are negligible for small timesteps, robotics simulators produce identical results for simulating smooth rigid body dynamics. However, robotics simulators can also support simulation of more complex kinematic mechanisms such as closed-loop constraints as well as various force or visual sensors. The need to model these components can exclude many simulators, as these features are not universally available.

Another important consideration when selecting simulators, especially in the context of robots interacting with their environment, is the choice of contact model. While the algorithms for smooth dynamics is consistent across robotics simulators, the choice of contact model to simulate the stiff contact dynamics can differ greatly. Simulators approximate the complex deformations using rigid body assumptions to improve computational performance. Truly rigid time-stepping contact models (Stewart and Trinkle, 2000) suffer from stiff numerics and do not capture the near-rigid nature of actual contact physics. Instead, simulators have opted for “soft" contact models justified by the above fact. However, these “soft" contact models take on many forms and thus are responsible for the primary differences in physics across simulators. Empirically (Acosta et al., 2022), both MuJoco and the TAMSI solver in Drake perform better than Bullet when capturing impact physics. For simulating frictional contacts, empirically we observe significant artifacts in MuJoCo such skipping effects when objects slide relative to each other at high speeds.

Finally, an overlooked simulation parameter for time-stepping simulators is the discretization timestep. While large timesteps reduce the computation burden, large timesteps can easily result in non-physical behavior such as the penetration of thin objects. However, even in non-pathological examples, large timesteps behave poorly with the regularized contact models, which leverage an assumption of small timesteps to mitigate these non-physical behaviors (Anitescu and Potra, 1997). Of the simulators discussed above and as of 2024, both Drake and MuJoCo utilize regularized contact models and thus are subject to these artifacts. In practice, we found a timestep of 1ms to be short enough capture the complex contact interactions presented in this thesis, while being large enough to enable real-time simulation.

3.4.2. Reflected Inertia

Typical commercial motors operate at low torques and high speeds compared to the high torques and relatively low speeds of legged robots and robot manipulators. For this reason, transmissions are used to alter the operating regimes. It is convenient from a modeling perspective to ignore the details of these transmissions as instead only consider the effective motor torque and speed at the output shaft. However, an often oversight is that the motor rotors themselves have inertia. Despite having relatively small actual inertias[3], the effective inertias of the rotors are non-negligible when viewed from the motor output shaft due to being mapped through the transmission. We use an approximation of the reflected inertia (Featherstone, 2014), which adds inertia contributions from the actuator rotor inertias directly to the corresponding diagonal terms of the mass matrix $M(q)$. Specifically we modify $M(q)$ at indices (ii) to be:

$$M_{ii} = M_{ii} + I_{RI,i} \tag{3.26}$$

$$I_{RI,i} = \rho_i^2 I_{rotor,i}, \tag{3.27}$$

where i are the indices of the actuated joints, $I_{RI,i}$ is the approximate reflected inertia, ρ_i is the gear ratio of the actuator transmission, and $I_{rotor,i}$ is the true rotor inertia of the actuator. The

[3]The hip roll and yaw motors on Cassie have rotor inertias of 61 $kg\,mm^2$ and the hip pitch and knee motors have rotor inertias of 365 $kg\,mm^2$. Source: https://github.com/agilityrobotics/cassie-doc/wiki/Abduction-and-Yaw-Motors.

magnitude of $I_{RI,i}$ relative to Mii for the actuated joints for both Cassie and the Franka Panda are listed in Table 3.2.

Table 3.2: Magnitude of reflected inertia compared to nominal values of specific elements of inertia matrix for Cassie and Franka. The reported inertia values are the diagonal elements of the inertia matrix corresponding to the actuated joints. The values of M_{ii} vary depending on the robot's configuration, while the values of $I_{RI,i}$ are fixed.

Robot	**Joint Name**	**Nominal Inertia** M_{ii} $(kg\,m^2)$	**Reflected Inertia** $I_{RI,i}$ $(kg\,m^2)$
Cassie	Hip Roll	0.664288	0.038125
	Hip Yaw	0.200439	0.038125
	Hip Pitch	0.548562	0.09344
	Knee	0.425824	0.09344
	Toe	0.00131458	0.01225
Franka	Joint 1	1.97368	0.605721
	Joint 2	0.817381	0.605721
	Joint 3	1.18889	0.462474
	Joint 4	0.844164	0.462474
	Joint 5	0.0251923	0.205544
	Joint 6	0.0264068	0.205544
	Joint 7	0.00181802	0.205544

3.4.3. Actuator Models

Another consideration for modeling actuators in simulation are actuation limits. Actuator limits are typically imposed as constant limits $|u| \leq u_{max}$, where u_{max} is set to the peak torques able to be achieved by the motors. In reality, the actual torque able to provided by the motors is also a function of the motor speeds. In slow quasi-static applications when the motors are moving slowly, this dependency is inactive and thus the constant limits are sufficient. However, when operating at faster speeds such as the running controller for Cassie, the torque-speed curve imposes relevant restrictions on the actual torque that can be achieved. To ensure we accurately account for this effect in our controllers when evaluating in simulation, we process the commanded control inputs through actuator model that acts as a filter:

$$u_{sat} = \text{clip}(u, -u_{limit}, u_{limit})$$
$$u_{limit} = 2u_{max}(1 - \frac{|v|}{v_{max}}),$$

where u is the commanded torque and u_{sat} is the actual torque sent to the simulated actuator. u_{max} is the peak torque and v and v_{max} are the current and max motor speeds respectively. While we could impose this torque-speed curve in our controllers, velocity measurements are occasionally unreliable and we found that avoiding these regimes altogether through controller design was more effective.

Another important consideration in actuator dynamics is the time delay between when sending a torque command and realizing that torque at the actuator. The time delay can be attributed to communication delay and also the time constant of the lowest level motor controllers. While there can be slight variations in both the communication delay as well as time constants, we model this delay in simulation as a constant time delay. From empirical measurements, we apply a delay of 6ms in our Cassie simulation when tuning for the closed loop performance.

3.4.4. Architecture Decisions for Sim-to-Real

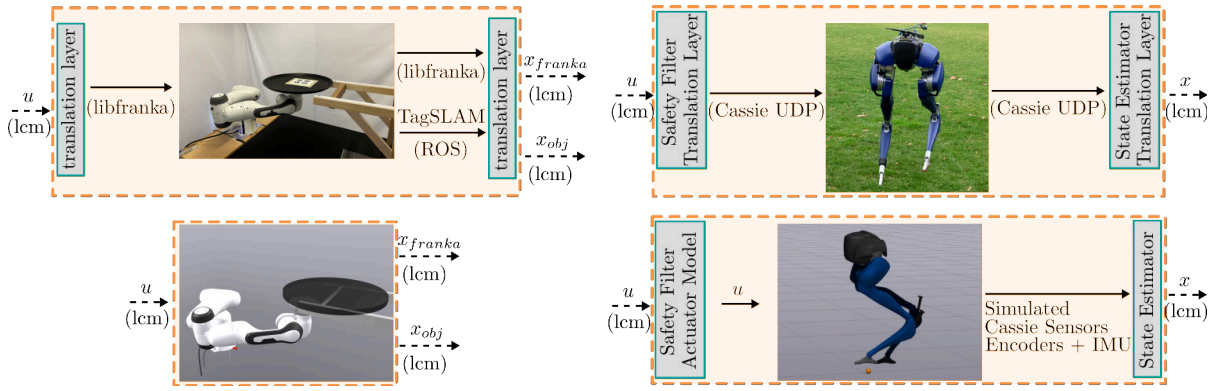


Figure 3.1: The translation layers are used to enable the simulated and physical robot to use the same communication abstraction, even across robot platforms.

We implement our controllers on two physical robots, the Cassie bipedal robot and the Franka Panda. Despite the differences in morphology, we control both robots using the OSC formulation described in Section 5.5.4, albeit with different models and constraints. To facilitate this, we introduce a communication abstraction, where the robot system takes in actuator torque commands u and outputs its measured state x. Furthermore, we implement translation layers to handle differences in the provided API, message type, and communication protocol. This enables a common control interface not only across robot platforms, but also between the simulated and physical robot

as illustrated in Fig. 3.1.

We use the inter-process communication framework, LCM (Huang et al., 2010), to separate all subcomponents of our controller stacks. The comprehensive list of subcomponents in our controller stack that we separate into separate processes are the state estimator, safety filter, low-level controller, high-level planner, simulator or real robot, and visualization. A key benefit of this separation is the ability to swap subcomponents such as the simulator or controller without needing to modify others. Including the visualization as a separate process allows for visualizing the estimated state of the real system through the same process for visualizing the simulated system.

Another key benefit is the ability to asynchronously run the controller and the simulator, which mimics the asynchronous nature of the real system and the controller. This is critical when designing controller frameworks for dynamic motions as delays, whether they are physical or digital, are a significant component of the closed-loop dynamics of the overall system. This modular approach ensures consistency when transferring from simulation to the real system; limiting the sim-to-real gap to solely a discrete number of modeling choices in the simulator.

CHAPTER 4

Impact-Invariant Control

Parts of this chapter were previously published as parts of William Yang and Michael Posa. Impact-Invariant Control with Applications to Bipedal Locomotion. In *IEEE/RSJ: International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic, 2021.
URL https://ieeexplore.ieee.org/abstract/document/9636094

In this chapter, we focus on the relevant uncertainties at an impact event. When a robot's foot makes contact with the world, the foot is brought quickly to a stop by a large contact impulse. Large contact forces and rapidly changing velocities hinders accurate state estimation. Coupled with the relatively poor predictive performance of our contact models (Halm and Posa, 2019; Fazeli et al., 2020; Remy, 2017), this combination of large state uncertainty and poor models makes control especially difficult. We argue that these uncertainties are impractical to overcome and instead we propose a general control framework that projects the tracking objectives down to a subspace where they are invariant to the impact event and therefore robust to just these uncertainties.

This chapter is ordered as follows. Section 4.1 introduces the robot models for the planar biped, Rabbit, and the 3D bipedal robot, Cassie. Section 4.2 introduces the concept of the impact-invariant subspace and how to apply it as a general modification to feedback control. Section 4.3 covers the metrics and baselines we use to evaluate our method, and the implementation details of the various controllers and results are reported in Sections 4.4 to 4.6. We conclude this chapter in Section 4.7 with a discussion about future directions.

4.1. Robot Models

We consider two legged robots, the five-link planar biped Rabbit (Chevallereau et al., 2003) and the 3D compliant bipedal robot Cassie shown in Fig. 4.2, to ground impact-invariant control in concrete applications. Both legged robots are modeled using conventional floating-base Lagrangian rigid-body dynamics.

Figure 4.1: Cassie is able to execute agile motions with non-negligible impacts like jumping (top), box jumping (bottom-left), and running (bottom-right) using impact-invariant control.

Rabbit has 7 degrees of freedom with 4 degrees of actuation while Cassie has 22 degrees of freedom with 10 degrees of actuation. Cassie has 4 physical leaf springs located at its ankle and knee joints. In our controllers, we treat these springs as rigid, reducing our controller model to 18 degrees of freedom. However when evaluating our results in simulation, we do include the springs, modeling them as torsional springs located directly at the ankle and knee joints.

4.2. Impact Invariance

In this section, we present the key ideas of this work. In Section 4.2.1, we explain the common challenges that arise when applying feedback during impacts. We then propose a solution in Section 4.2.2, which also introduces the concept of the impact-invariant subspace. In Section 4.2.3 and Section 4.2.4, we explain the practical details of how to implement the subspace in the context of an operational space controller. Finally, in Section 4.2.5 we derive our formulation for impact-invariant control from the perspective of robust optimal control.

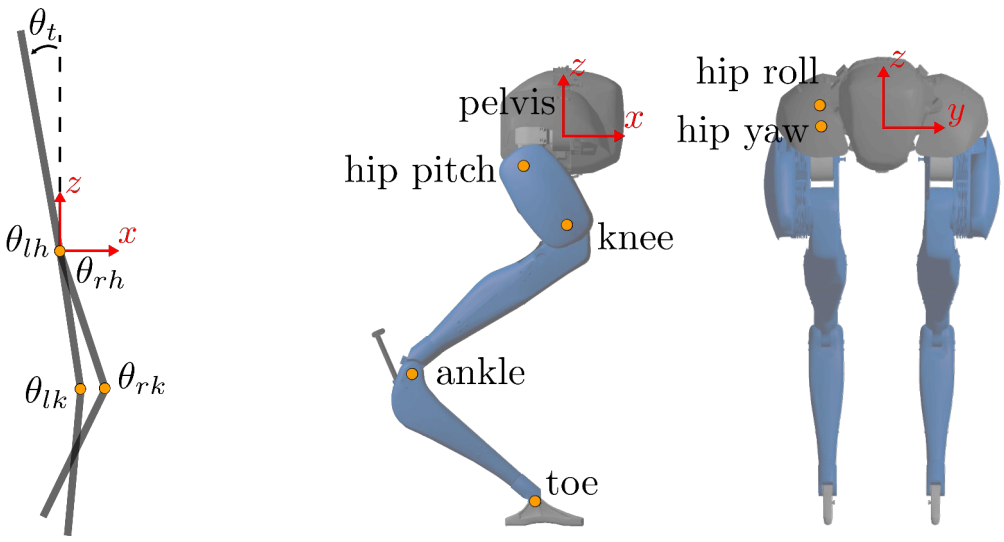


Figure 4.2: We use both the planar biped Rabbit (left) and the 3D compliant bipedal robot Cassie (right) as concrete examples to highlight the advantages of impact-invariant control.

4.2.1. Challenges of Control During Impacts

To motivate the concept of the impact-invariant subspace, we begin by highlighting and describing the difficulties of applying feedback control during an impact event. For the sake of simplicity, we consider a feedback controller with constant feedback gains that controls an output $y : \mathbb{R}^{n_q} \to \mathbb{R}^{n_y}$ to track a time-varying trajectory $y_{des}(t) : [0, \infty) \to \mathbb{R}^{n_y}$ by driving the tracking error $\tilde{y}(t) = y_{des}(t) - y(t)$ to zero. This is commonly accomplished with control law $u = u_{ff} + u_{fb}$ where u_{ff} is the feedforward controller effort required to follow the reference acceleration $\ddot{y}_{des}$ and u_{fb} is the PD feedback component given by:

$$u_{fb}(t) = K_p \tilde{y}(t) + K_d \dot{\tilde{y}}(t). \tag{4.1}$$

The reference trajectory $\dot{y}_{des}(t)$ for systems that make contact with their environment has discontinuities at the impact events in order to be dynamically consistent with (3.8). Therefore, in a short time window around an impact event, there will be a discontinuity in the reference trajectory $\dot{y}_{des}(t)$ at the nominal impact time and a near-discontinuity in the actual robot state when the system $\dot{y}(t)$ makes contact with the ground as shown in Fig. 4.3. Because the robot configuration

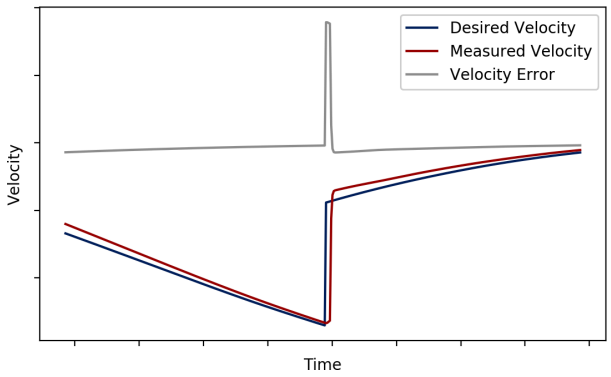


Figure 4.3: Illustration of a system that undergoes an impact event. The desired velocity plan correctly includes the discontinuity as predicted by rigid body impact laws and the measured velocity is being properly regulated to match the desired plan. However, due to the mismatch in impact time, the velocity error inevitably spikes during the impact event.

is approximately constant over the impact event, the change in controller effort is governed by the change in velocity error:

$$\Delta u \approx K_d \Delta(\dot{y}_{des} - \dot{y}), \tag{4.2}$$

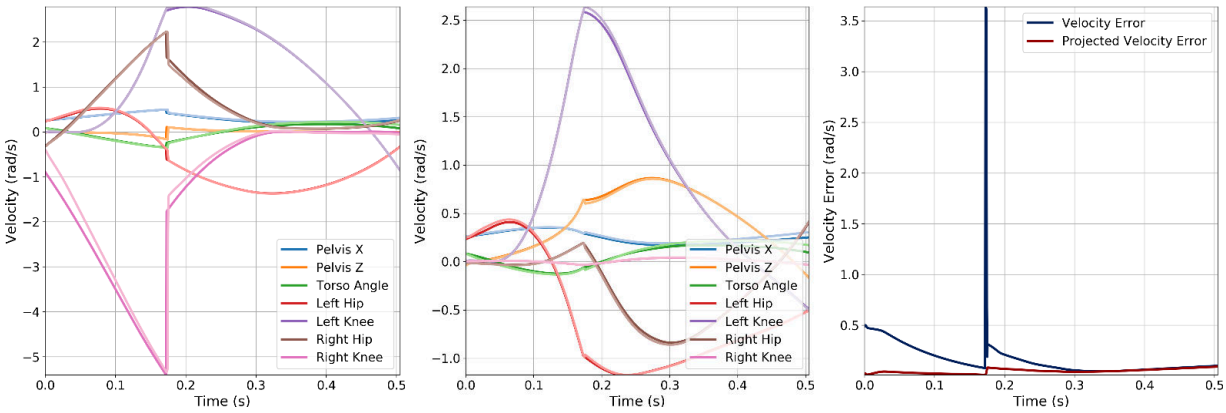
thus any mismatches in impact timing will unavoidably result in spikes in the feedback error signal, which has been similarly noted in (Rijnen et al., 2017).

Remark 1. *The previous example assumes the jump in the reference trajectory is time-based. Although it is possible to formulate trajectories with event-triggered jumps, these methods require detection, which for state-of-the-art methods still have delays of 4-5 ms (Bledt et al., 2018a). Moreover, in reality, impacts are not resolved instantaneously but rather over several milliseconds to tens of milliseconds. In this time span, it is not clear which reference trajectory to use, as using either trajectory will output a large tracking error.*

Note that a large tracking error, shown in Fig. 4.3, results from only a small difference in impact timing, yet the controller will respond to the large velocity error and introduce controller-induced disturbances. Furthermore, this sensitivity to the impact event is amplified by the large contact forces that impair state estimation and result in likely inaccurate velocity measurements due to

necessary filtering.

The key challenges of applying feedback during impacts can thus be summarized as: impacts are brief moments of high uncertainty where our references are poorly defined and our measurements are inaccurate.



(a) Generalized velocities for a periodic walking trajectory for a planar five-link biped.

(b) Generalized velocities projected to the impact-invariant space

(c) Comparison of tracking errors in the two spaces

Figure 4.4: The velocities are shown for a periodic symmetric walking trajectory (a) for a planar five-link biped. Because the trajectory is constructed to be symmetric, only half of the gait is shown without loss of information. The instantaneous jump in velocities is due to an impact event when the right foot makes contact with the ground. The darker shade indicates the nominal trajectory while the lighter shade is an example of actual velocities when tracking to the nominal trajectory. Despite the qualitatively "good" tracking of the velocities, the error near the impact event still exhibits a sharp jump (c) due to the discontinuity from impact. The same velocities projected to the impact-invariant subspace (b), do not have experience this discontinuity and therefore results in a much smaller tracking error, which is a better reflection of the actual system tracking.

4.2.2. Impact-Invariant Subspace

The key insight in resolving the problem of control during impacts is inspired by (Gong and Grizzle, 2022), in which Gong and Grizzle delineate desirable properties of angular momentum about the contact point. They highlight that it is invariant over impacts on flat ground, meaning that it is continuous over the impact event despite it being a function of velocity.

The concept of an impact-invariant subspace is a generalization of this property. We observe that there is a space of velocities that, like angular momentum about the contact point, are continuous through impacts for *any* contact impulse. By switching to track only these outputs in a small time

window around anticipated impacts, we avoid controller-induced disturbances from uncertainty in the impact event. While angular momentum is in $\mathbb{R}^3$ or $\mathbb{R}^2$ for planar systems such as Rabbit, the impact-invariant subspace is in $\mathbb{R}^{n_v - n_c}$, where n_v is the dimension of generalized velocities and n_c is the dimension of the contact impulse of the impact event. For Rabbit walking, this space is in $\mathbb{R}^{7-2}$. For Cassie, each foot on the ground imposes a contact constraint of dimension 5 and the four bar linkage on each leg provides a distance constraint of dimension 1 that is always active [4]. Therefore the impact-invariant subspace is in $\mathbb{R}^{18-7}$ for impacts with a single foot (walking, running) and is in $\mathbb{R}^{18-12}$ for impacts with both feet (jumping). The implication of the impact-invariant subspace is a space that provides full robustness to uncertainty caused by impacts, while minimally reducing control authority. This results in better tracking which is important for precise dynamic motions.

The impact-invariant subspace is defined as the nullspace of $(M^{-1}J_\lambda^T)^T$ or left nullspace (Strang, 2022) of $M^{-1}J_\lambda^T$, where J_λ again is the Jacobian for the active constraints. Thus a basis $P(q) \in \mathbb{R}^{(n_v - n_c) \times n_q}$ for this nullspace is such that:

$$PM^{-1}J_\lambda^T \Lambda = 0. \tag{4.3}$$

The velocities during the impact event can be described as $v = v^- + M^{-1}J_\lambda^T \Lambda$, where here Λ represents the contact impulse experienced thus far. The velocity during impact in this basis remains the same as the pre-impact velocity, $Pv = P(v^- + M^{-1}J_\lambda^T \Lambda) = P(v^-)$, which is the intended effect. That is, for any contact impulse Λ, the velocities in the impact-invariant subspace are unchanged. To project the generalized velocities down to the impact-invariant subspace, we can create an orthonormal projection matrix $Q(q) \in \mathbb{R}^{n_v \times n_v} = P^T P$. In practice, we can compute this projection matrix as $Q = I - M^{-1}J_\lambda^T (J_\lambda M^{-T} M^{-1} J_\lambda^T) J_\lambda M^{-T}$, which we use to define the projected velocity error as $Q(v_{des} - v)$.

Let's consider a concrete example of how the impact-invariant subspace can be applied to control of a legged robot using a walking gait of a simulated five-link biped. We solve a hybrid trajectory

[4] We model Cassie's feet as two point contacts on the same rigid body, with each point imposing a constraint of dimension 3. One dimension is redundant, thus resulting in an overall active contact constraint dimension of 5.

optimization problem (Posa et al., 2016) to generate a periodic walking gait for the planar biped that satisfies the rigid body impact law (3.9). The generalized velocities for half of gait are shown in Fig. 4.4a, where the jump in velocities is a result from the right foot of the robot making impact with the ground. As noted in the figure caption, the darker shade indicates the nominal or target velocities, while the lighter shade indicates an example of velocities the robot may experience when tracking to the nominal trajectory. For these same velocities, we can compute and apply the impact-invariant projection to arrive at the projected velocities shown in Fig. 4.4b. An important observation of the projected velocities is that they are now continuous over the impact event, which eliminates the large spike in tracking error that was a result of tracking a discontinuous signal as shown in Fig. 4.4c. Notice, although every degree of freedom experiences a jump in its velocities at the impact event, the projected velocities not zero. In fact, the original and projected velocities of the left knee joint are remarkably similar, which can be intuitively understood by the fact that forces that left knee is only distantly connected to forces that enter through the right foot according to the kinematic tree. In contrast, the projected velocities of the right knee joint are practically zero.

Similarly, to illustrate the benefit on a *physical robot*, the joint velocities for Cassie executing a jumping motion right when it lands across 8 experiments are shown in Fig. 4.5. Details of the jumping controller and experiments are given in Section 4.5. Observe that the projected joint velocities are significantly smoother than the original joint velocities.

4.2.3. Implementation for Task Space Tracking

The objective of impact-invariant control is to apply control on the subspace where the changes in the velocity introduced via the impact map, $M^{-1}J_\lambda^T$, do not appear. In practice, we accomplish this by modifying our controllers to eliminate the component of the velocity error that is within the subspace spanned by the impact map $M^{-1}J_\lambda^T$ by solving the following optimization problem:

$$\min_{\lambda} \quad \left\| \dot{y}_{des} - J_y(v + M^{-1}J_\lambda^T \lambda) \right\|_2 \tag{4.4}$$

$$\text{s.t.} \quad J_h(v + M^{-1}J_\lambda^T \lambda) = 0. \tag{4.5}$$

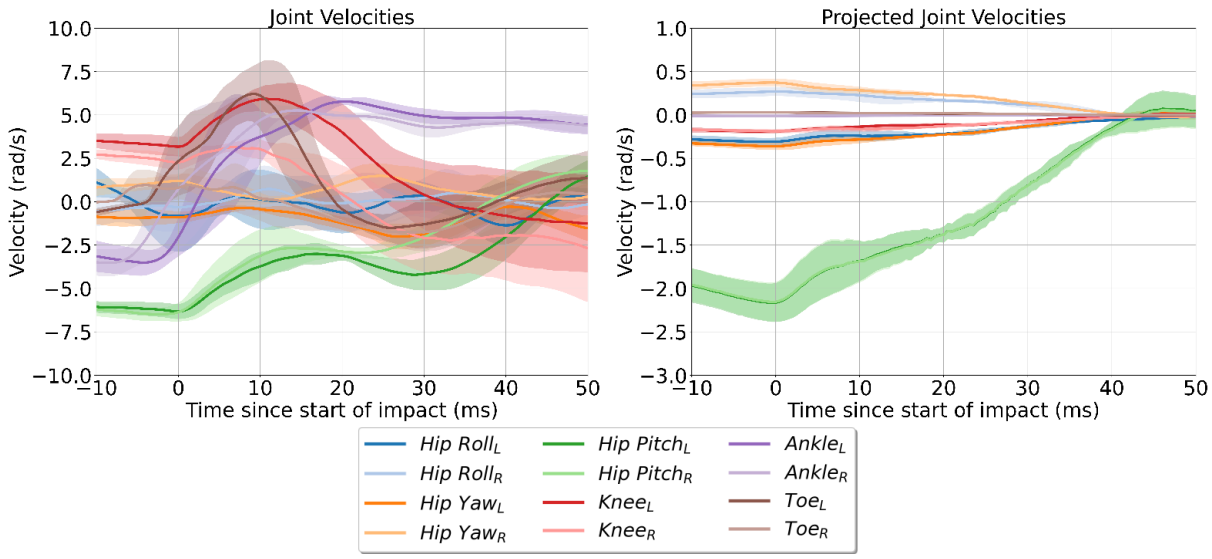


Figure 4.5: Demonstration of the impact-invariant projection on joint velocity data from 8 consecutive jumping experiments on the physical Cassie robot. Joint velocities (top) during the landing event change rapidly. By projecting the same joint velocities to the impact-invariant subspace (bottom), the values are more consistent and more amenable for feedback control. Note, the change in joint velocities primarily occurs within a time span of only 10 - 20 ms. The L and R subscripts indicate the left and right leg respectively.

Here, we overload J_y to represent the matrix constructed by stacking the active output space Jacobians $J_{y,i} \forall i$. J_h is the Jacobian for the holonomic constraints that are unambiguously active during impact [5].

This reduces the total error by subtracting a correction term $M^{-1}J_\lambda^T\lambda$ that by construction is in the range of $M^{-1}J_\lambda^T$. We must include the constraint forces for all active constraints, as contact forces from the impact event will cause corresponding reaction forces, which should still satisfy kinematic feasibility of the velocities enforced by (4.5). Notice that this optimization problem is an equality constrained QP, which we are able to solve this in closed form:

$$\begin{bmatrix} \lambda^* \\ \mu^* \end{bmatrix} = \begin{bmatrix} A^TA & B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} A^T(\dot{y}_{des} - \dot{y}) \\ J_h v \end{bmatrix}, \tag{4.6}$$

[5]Unambiguously active constraints refer to constraints that are active both before and after the impact event. For example, the four-bar linkage constraint is always active and for walking with a double stance phase, the contact constraint for the current stance foot is active both before and after the other foot makes contact. To be explicit, J_h is a subset of J_λ because J_λ includes all active constraints of the current impact event.

where $A = J_y M^{-1} J_\lambda^T$, $B = J_h M^{-1} J_\lambda^T$, and μ^* is the Lagrange multiplier for the holonomic constraint.

Applying λ^* back into the OSC formulation, we combine the correction and the measured velocity to define the projected generalized velocities as:

$$v_{proj} = v + M^{-1} J_\lambda^T \lambda^*. \tag{4.7}$$

We then define the projected output velocity, $\dot{y}_{proj}$, and projected output space velocity error, $\dot{\tilde{y}}_{proj}$, as:

$$\dot{y}_{proj} = J_y v_{proj} \tag{4.8}$$

$$\dot{\tilde{y}}_{proj} = \dot{y}_{des} - \dot{y}_{proj}. \tag{4.9}$$

The constrained least-squares problem (4.4) used to modify any general task-space velocity error to be impact-invariant is a generalization of the projection matrix Q from (4.3). We show that (4.4) performs the same modification to the tracking error as Q in Section 4.8. The projected output space velocity error $\dot{\tilde{y}}_{proj}$ is then used in the OSC feedback law (3.17) to create the impact-invariant feedback law:

$$\ddot{y}_{cmd} = \ddot{y}_{des} + K_p(y_{des} - y) + K_d(\dot{y}_{des} - \dot{y}_{proj}). \tag{4.10}$$

Stacking the output space Jacobians to construct a single J_y ensures only a single λ^* is used to project the tracking error. Note, the derivative gains and weighting matrices defined in (3.17) (3.18) can intuitively be included when constructing A in (4.6), but in practice we did not find a noticeable effect from including them.

Constraints on the Projection

Due to the lack of constraints on λ, the projection impulse is not guaranteed to be physically possible. λ could be constrained to lie within the friction cone FC. However, upon further examination, inclusion of these constraints may be undesirable. This is because sensitivity to the impact event can result from the *absence* of expected impacts as well - consider the case when the robot makes contact after the nominal impact time. Constraining $\lambda \in FC \cup -FC$ is not practical as this set is non-convex. Although it is possible to formulate this as a binary mixed integer program, with a integer variable per point contact, solving this problem was considered to require too many assumptions to justify the additional complexity.

4.2.4. Activating the Projection

The benefit of explicitly computing the error correction $M^{-1}J_\lambda^T\lambda^*$ is that it becomes trivial to smoothly blend in the projection.

Because impact-invariant control essentially ignores errors in the space of impacts, we should only use it when we anticipate impacts. In practice, we do this by only activating the projection in a time window near anticipated impact events. To avoid introducing discontinuities when activating the projection, we blend in the correction, $M^{-1}J_\lambda^T\lambda^*$ using a blending function $\alpha(t)$ visualized in Fig. 4.6 constructed using a sigmoid function $\sigma(x) = \frac{e^x}{1+e^x}$.

$$\alpha(t) = \begin{cases} \sigma(\frac{T-|t-t_s|}{\tau}) & t_s - 1.5T \leq t \leq t_s + 1.5T \\ 0 & otherwise \end{cases} \tag{4.11}$$

where t is the current time, t_s is the switching time defined as the nominal impact time of the reference trajectory, T is the duration of the projection window, and τ is the time constant that determines the rate of the blending.

We then use $\alpha(t)$ to modify (4.7) to be:

$$v_{proj} = v + \alpha(t)M^{-1}J_\lambda^T\lambda^*. \tag{4.12}$$

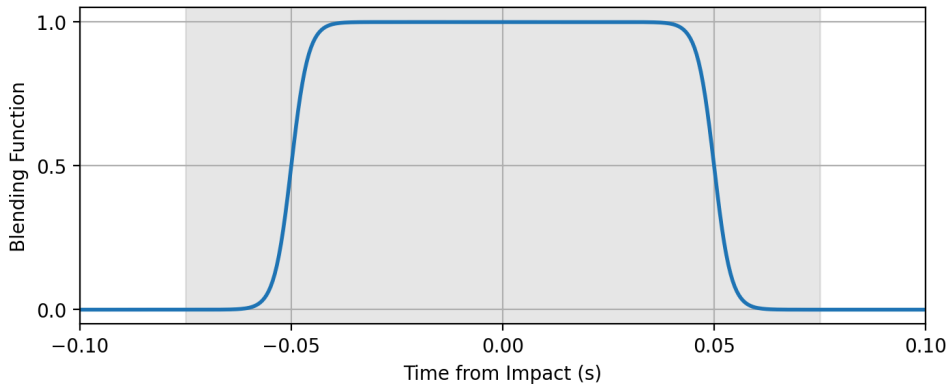


Figure 4.6: Blending function for the impact-invariant projection for a window T of 50 ms and time constant τ of 2 ms.

Note our choice for the blending function $\alpha(t)$ is arbitrary; any monotonic continuous function with a range of $[0, 1]$ can accomplish a similar purpose. Additionally, while $\alpha(t)$ is a function of time, we can also activate/blend in the impact-invariant projection purely as a function of the robot's state. For example, we can use the distance between the foot or end effector and the environment in place of $t - t_s$.

4.2.5. Robust Optimal Control Perspective

We can view impact-invariant control from the perspective of robust control, which formulates a min-max optimization problem (Salmon, 1968) that minimizes a control objective with subject to worst-case disturbances. Our control objective is to minimally perturb our control inputs, ν, as is commonly done in the control barrier functions (Ames et al., 2019), where we take these inputs to be an intermediary value: the tracking error of our tracking objectives $\nu^* = \dot{y}_{des} - J_y v$, where $\nu \in \mathbb{R}^{n_y}$. The disturbance is the unknown contact impulse Λ, which causes the measured velocity $\hat{v}$ to differ from the true velocity according to $\hat{v} = v + M^{-1} J_\lambda^T \Lambda$. Because we can only measure $\hat{v}$, our control input is:

$$\nu = \dot{y}_{des} - J_y(v + M^{-1} J_\lambda^T \Lambda). \tag{4.13}$$

However, since Λ is unknown and potentially quite large, the controller (4.13) is brittle. We formulate the following minimally perturbing robust controller, where robustness to arbitrary Λ enters as

the constraint that the controller must not be sensitive to Λ:

$$\min_{\nu} \max_{\Lambda} \quad \|\nu - \nu^*\|_2^2 \tag{4.14}$$

$$\text{s.t.} \quad \frac{\partial \nu}{\partial \Lambda} = 0. \tag{4.15}$$

If we restrict ν to be a linear function of the estimate $\nu(\hat{v}) = \dot{y}_{des} - J_y\hat{v}$, we reparameterize $\nu(\hat{v})$ as $\nu(\hat{v}) = Q(\dot{y}_{des} - J_y(\hat{v}))$, where $Q \in \mathbb{R}^{n_y \times n_y}$. Under this parametrization, the robustness constraint can be simplified to $QJ_yM^{-1}J_\lambda^T = 0$ and we arrive at the following matrix optimization problem:

$$\begin{aligned} \min_{Q} \quad & \|(Q - I)(\dot{y}_{des} - J_y v)\|_2^2 \\ \text{s.t.} \quad & QJ_yM^{-1}J_\lambda^T = 0. \end{aligned}$$

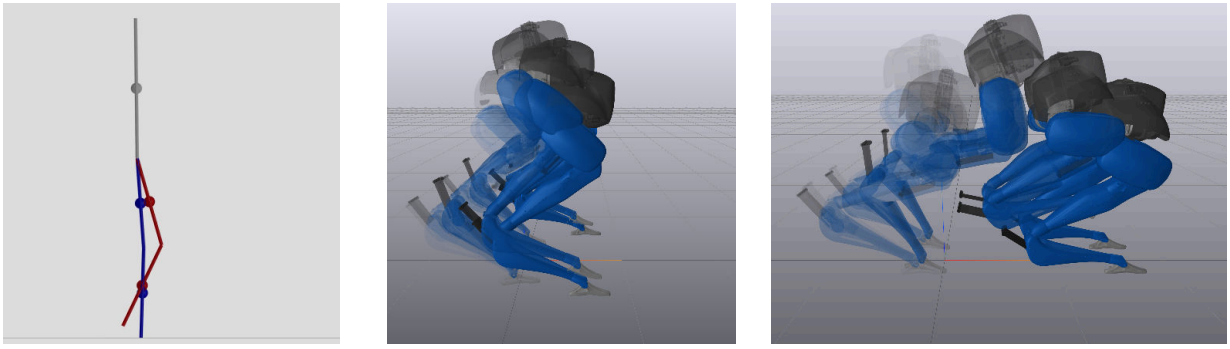
We cannot measure the true velocity v in the expression $\dot{y}_{des} - J_y v$, so we instead minimize the matrix norm induced by the vector 2-norm, which is $\|Q - I\|_2 = \sup_{\dot{y}_{des} - J_y v \neq 0} \frac{\|(Q-I)(\dot{y}_{des} - J_y v)\|_2}{\|\dot{y}_{des} - J_y v\|_2}$. Minimizing this matrix norm is a semi-definite program (SDP) (Boyd and Vandenberghe, 2004). It is not currently feasible to solve this SDP at real-time control rates, so we instead minimize an upper bound on the robust minimum-perturbation cost. We substitute the 2-norm with a Frobenius norm and arrive at:

$$\begin{aligned} \min_{Q} \quad & \|Q - I\|_F^2 \\ \text{s.t.} \quad & QJ_yM^{-1}J_\lambda^T = 0, \end{aligned} \tag{4.16}$$

which is equivalent to solving for a projection matrix for the left-nullspace of $J_yM^{-1}J_\lambda^T$, which we show is equivalent to (4.4) in Section 4.8.1. Thus we have shown that projecting the velocities to the null-space of the impact map is the optimal linear policy, as formulated by (4.14) (4.15). Note, in place of using velocity tracking error as our control inputs, we could instead consider the final control effort, $\nu = u$, as in (Ames et al., 2019), for example. The projection would then depend on the nominal controller itself, via gains K_d and weights W_i as well as the system dynamics (3.19).

These scaling terms may slightly modify the solution, but as mentioned in Section 4.2.3, we did not find a noticeable effect in simulation experiments.

4.3. Experiments and Evaluation



(a) Walk (b) Jump (c) Long jump

(d) Box jump (e) Down jump

Figure 4.7: Reference trajectories generated using full model trajectory optimization.

To showcase the advantages of using the impact-invariant subspace, we apply the aforementioned projection on the following examples, each organized in the following sections:

- In Section 4.4, we consider, in simulation, a joint-space inverse dynamics walking controller for the planar five-link biped Rabbit (Chevallereau et al., 2003), we compare impact-invariant control to a default controller that makes no adjustment near impacts and to a controller that applies no-derivative feedback near impact.

- In Section 4.5, we formulate an operational space jumping controller for the 3D bipedal robot Cassie. We consider a basic jump as well as more dynamic jumps such as a long jump, box jump, and down jump. In simulation, we evaluate the long jump and down jump controllers

through a parameter study. On hardware, we validate the jump and box jump controllers with impact-invariant control.

- Section 4.6, we develop an operational space running controller for Cassie, which we evaluate in simulation and on hardware.

4.3.1. Baseline Comparisons

We include three controllers in our evaluations:

- The **default controller** is a controller that makes no special considerations with regards to the impact event other than switching contact modes at the nominal impact time. In other words, the standard hybrid controller.

- The **no derivative feedback** controller is a controller that sets all derivative gains to zero ($K_d = 0$) in a window before and after the nominal impact time. For proper comparisons, we use the same window between the no derivative feedback controller and the impact-invariant controller.

- The **impact-invariant projection** controller is our proposed method. We blend in the projection in a window before and after the nominal impact time.

The **default** and **no derivative feedback** controllers serve as the baselines for our comparisons. They represent either side of the extremes, where the default controller is extremely sensitive to uncertainty in the impact event but relinquishes no control authority, while the no derivative feedback controller gives up entirely on tracking velocities during impacts but loses significant control authority. Although eliminating derivative feedback near impacts may seem extreme, the intermediate mode described in (van Steen et al., 2023) is exactly this choice of controller.

4.3.2. Evaluation Metrics

We evaluate the three controllers (default, no derivative feedback, impact-invariant) on three metrics of decreasing importance: stability, tracking performance, and controller effort.

In this paper, we define stability as a binary value of whether or not the robot falls when executing the motion. We consider stability to be the primary metric, as falling for legged robots is often considered catastrophic failure. Minimizing tracking error is the objective of our controllers as it is essential for stability, so naturally it is our second most important evaluation metric. However, a key insight of our method is that there is uncertainty in both our target state and our measured state during impacts, and thus the standard measure of tracking error, particularly in the velocities, may not necessarily be a useful metric. For these reasons, we selectively report what we consider are the relevant tracking errors for each experiment. Finally, for the jumping experiments, the whole body motion is significantly more sensitive to the initial leap than to the control actions taken during the impact event when the robot lands. For this reason, tracking error is a poor metric to compare how the robot performs near the impact event, which is the focus of this paper. Instead, we evaluate the controller performance using the control efforts taken near the impact event, considering large controller spikes to be undesirable as those are prone to damaging the robot.

4.4. Five-Link Biped Walking Controller

We showcase the benefits of our method on a joint-space inverse dynamics controller for the planar five-link biped. This simple system and controller serves as a controlled example to demonstrate the directional nature of the impact-invariant projection.

4.4.1. Experimental Setup

In simulation, we model our planar five-link biped using the length, mass and, inertia values of Rabbit specified in (Chevallereau et al., 2003), replicated in Table 4.5 for completeness. The joint-space controller is tasked with tracking the hip and knee joint angles in both legs for a total of 4 tracking objectives to match the degrees of actuation.

We generate a periodic walking trajectory using trajectory optimization and perturb the swing foot vertical velocity by 0.1 m/s at the start of the trajectory, which causes the robot to makes contact approximately 5ms after the nominal impact time.

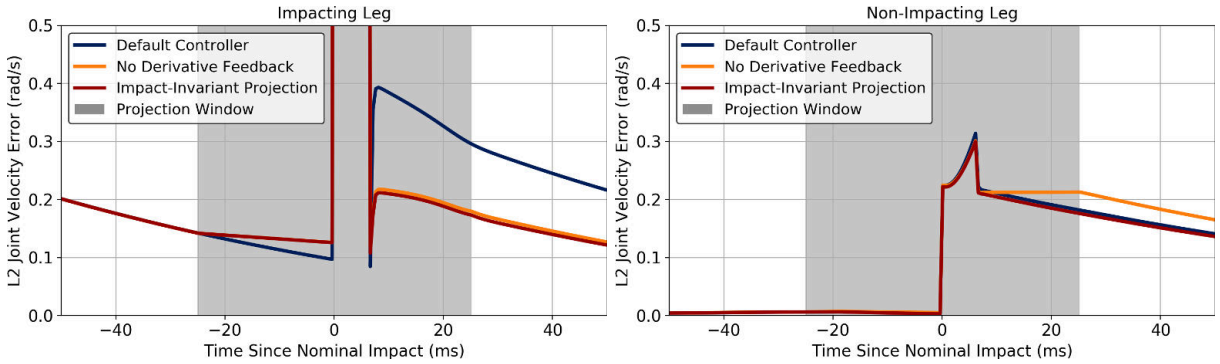


Figure 4.8: The joint velocity tracking errors are shown for the impacting leg (left) and the non-impacting leg (right) for three strategies. The controller that utilizes the impact-invariant projection is shown to be robust to the mismatch in impact timing as evidenced by lower tracking error compared to the default controller. The impact-invariant controller is also able to maintain full control authority over the joints in the non-impacting leg compared to the controller that applies no derivative feedback in the same window. The time window where no-derivative feedback and the impact-invariant projection are active is shown in grey.

4.4.2. Results

The tracking errors for the joints in the impacting and non-impacting leg are shown in Fig. 4.8. The impact-invariant controller is able to achieve the best tracking performance out of the three controllers for *both* legs. It has better tracking performance than the default controller for the joint velocities of the impacting leg. At the same time, it has better tracking performance than the controller with no derivative feedback for the joint velocities of the non-impacting leg by appropriately regulating the velocities in those joints.

4.5. Cassie Jumping Controller

Next, we evaluate the performance of the impact-invariant projection on a jumping controller for Cassie. We chose to look at jumping due to the richness of the impact event: the robot cannot accurately estimate its state (Jeon et al., 2022) when it is in the air and must make impact with the ground with non-zero velocity.

4.5.1. Controller Formulation

We formulate the jumping controller as a trajectory tracking problem, where the target trajectory is generated using the full order model and translated to task-space trajectories which are tracked

by an operational space controller.

Reference Trajectories

The reference jumping trajectories were computed offline by solving a constrained trajectory optimization problem on the full rigid body model of Cassie, excluding the springs. While the springs are a significant component to the dynamics, we found that including them made solver convergence extremely difficult.

The various jumping trajectories are distinguished by the constraints imposed:

- The normal jump trajectory was constrained to have the robot pelvis reach an apex height of 0.15 m above its initial starting height and to have both feet reach 15 cm of clearance above the ground at the apex.
- The long jump trajectory has the feet land 0.7 m ahead of their initial position.
- The box jump trajectory has the feet land on a box that is 0.5 m tall and 0.3 m in front of the starting configuration.
- The down jump trajectory has the feet land on a platform 0.5 m below and 0.3 m in front of the starting configuration.

The trajectory optimization problems were transcribed using DIRCON (Posa et al., 2016) and solved using a combination of IPOPT (Wächter and Biegler, 2006) and SNOPT (Gill et al., 2005). Traces of the reference trajectories are shown in Fig. 4.7. The same jumping trajectories were used in both simulation and on the physical robot.

Finite State Machine

We decompose the jumping motion into three states CROUCH, FLIGHT, LAND, and switch between states at fixed times as computed from the reference trajectories.

Table 4.1: Tracking objectives for the jumping controller. All values are expressed in the world frame.

Symbol	Description
$r \in \mathbb{R}^3$	Pelvis position
$\psi \in SO(3)$	Pelvis orientation
$y_L, y_R \in \mathbb{R}^3$	Foot position relative to pelvis
$\beta_L, \beta_R \in \mathbb{R}$	Hip yaw angle
$\phi_L, \phi_R \in \mathbb{R}$	Toe joint angle

Tracking Objectives

The tracking objectives are listed in Table 4.1. We define tracking objectives such as the foot and pelvis position relative to other points on the robot to reduce sensitivity to errors in state estimation. During the CROUCH and LAND states, the active objectives are $[r, \psi]$. During the FLIGHT state, the active objectives are $[y_L, y_R, \beta_L, \beta_R, \phi_L, \phi_R]$. The active tracking objectives per mode are also illustrated in Fig. 4.9. Similar outputs are used in another jumping controller for Cassie (Xiong and Ames, 2018).

4.5.2. Simulation Results

Timing the switch from the FLIGHT state to the LAND state is critical to stabilize the jump. Impact-invariant control reduces sensitivity to the impact timing and thus enables a greater margin of error. To evaluate this effect, for all the jumping motions we perturb the transition time t_s from the nominal switching time by $[-0.025s, 0.025s]$ and evaluate a range of projection duration $T \in [0.0s, 0.1s]$ to empirically evaluate the sensitivity of the controller.

In addition to testing whether the robot successfully controls the jump, we also evaluate the overall control input cost:

$$J_u = \int_{t_0}^{t_f} u^T W_u u dt, \tag{4.17}$$

where W_u is the same regularization weight on u we use in the OSC. 5 experiments per pair of transition times and projection durations are evaluated using the Drake simulator. The results are

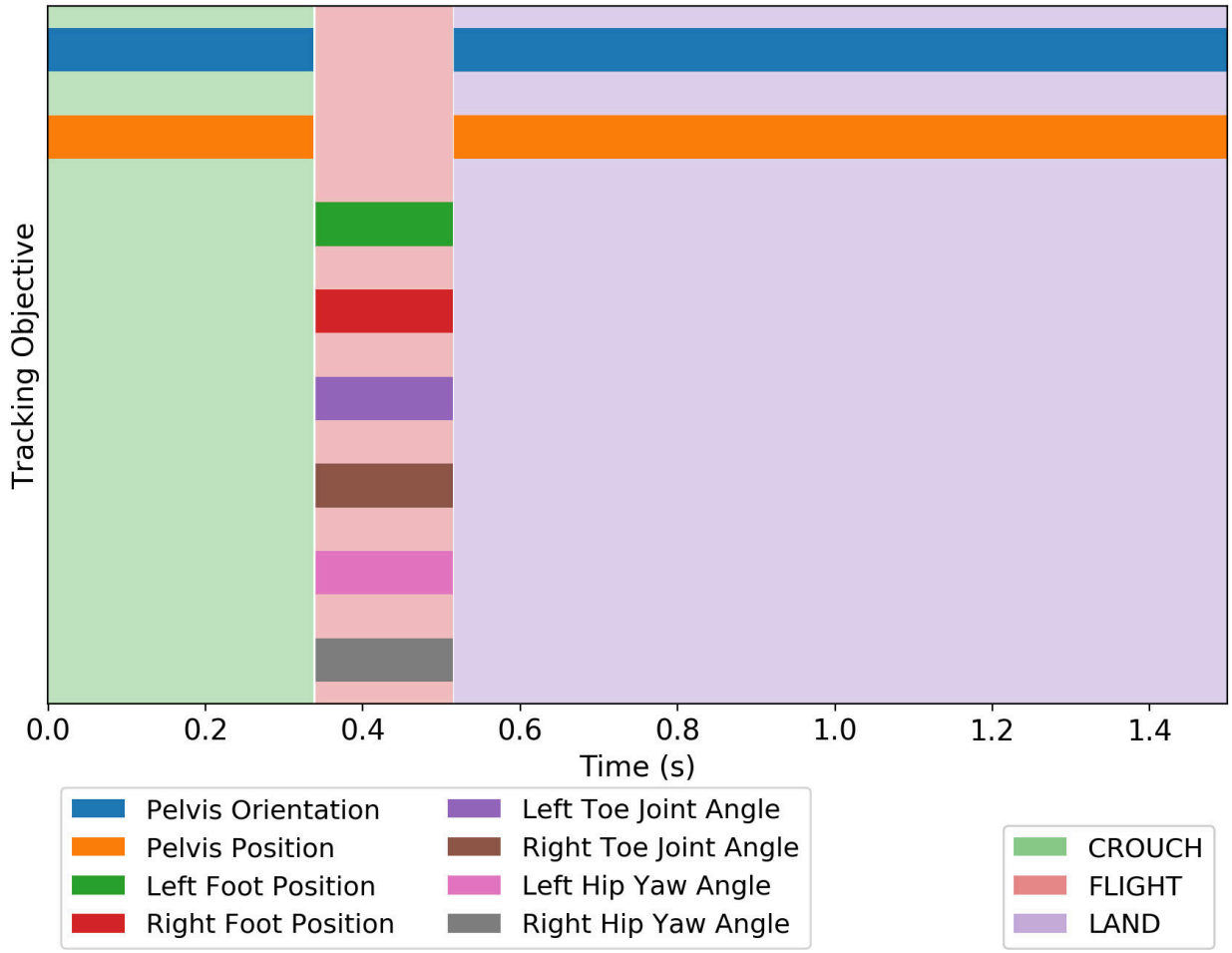
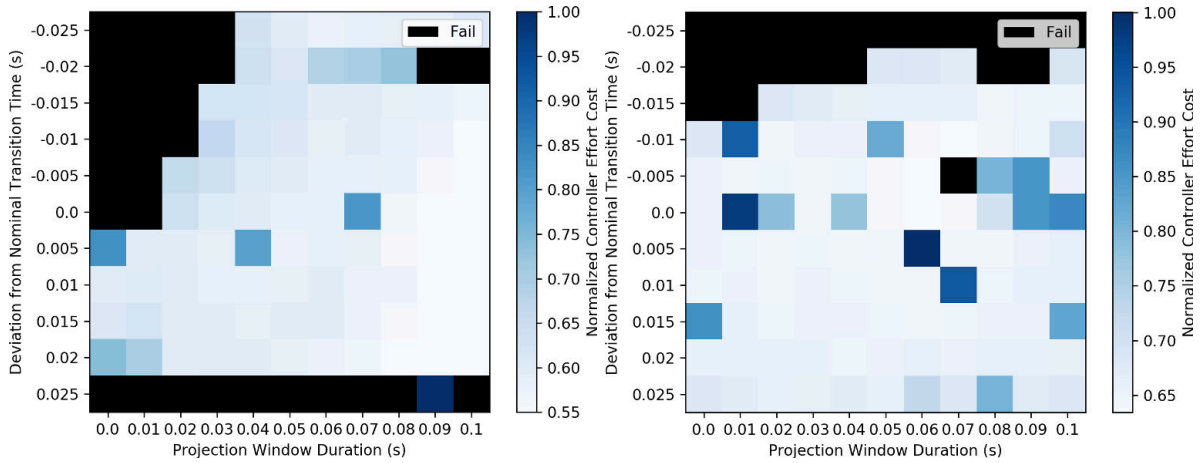


Figure 4.9: Active tracking objectives per mode for the jumping controller executing the jump trajectory.

reported in Fig. 4.10, with J_u normalized so that the largest value is 1. We exclude cases when there is more than one failure to avoid saturating the cost metric.

The region of stable jumps increases as the projection window duration increases, while there is not a significant effect on the input cost. We see a more noticeable improvement from the impact-invariant controller in the long jump than the down jump. We theorize that this is because the long jump motion is more difficult to stabilize, most likely from less control authority due to friction code limits, and therefore the improved robustness from the impact-invariant controller has a greater marginal effect.



(a) Long jump (b) Down jump

Figure 4.10: Simulation effort costs for (a) long jump and (b) down jump evaluated over a range of fixed transition times and projection window durations. Five trials are evaluated for each pair of transition times and projection durations. The controller without the impact-invariant projection corresponds to a projection window duration of 0.0 s. The regions where the robot fails to stabilize for over half the trials are marked as “Fail". As the projection window increases, the range of successful transition times increases. This effect is more pronounced for the long jump.

4.5.3. Hardware Results

Experiments using the controller with and without the impact-invariant projection were both consistently able to successfully complete the basic jump. Snapshots of the jumping motion are shown in Fig. 4.1. As seen in Fig. 4.5, the joint velocities change rapidly during the impact event. By projecting the velocity error of the outputs (position and orientation of the pelvis) to the impact-invariant subspace, we avoid overreacting to these rapid velocity changes in a principled manner. The effects of this can be seen in Fig. 4.11, where the change in knee motor efforts, particularly at the impact event, are significantly reduced when using the impact-invariant projection. We choose to show the knee motor efforts because they exhibit the largest change at the impact event due to their role in absorbing the weight of the pelvis at impact. This smoothing out of the commanded motor efforts is similar to what we see in simulation. The jumping motions for the controller both with and without the impact-invariant projection are included in the supplementary video.

Among the other jumping trajectories, we chose to test the box jump trajectory on hardware as it is least likely to damage the robot. We positioned 16 in (∼0.4 m) tall wooden boxes in front of the

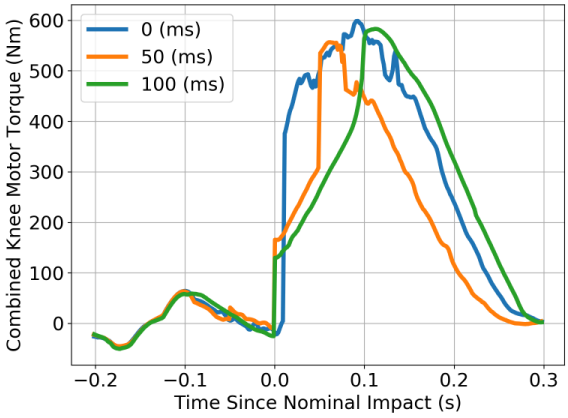


Figure 4.11: Motor efforts on Cassie executing the default jumping motion. The combined knee motor torques commanded by the jumping controller are shown for three different durations of the impact-invariant projection.

robot and executed the same tracking controller using a 50 ms projection duration. Although the reference jumping trajectory we optimized was for a box height of 0.5 m, the robustness afforded by the impact-invariant controller enabled the controller to successfully achieve the jump. A frame of the controller executing the box jump is shown in Fig. 4.1 and is also included in the supplementary video.

Approximately 20 trials were conducted to tune the controller parameters in order to achieve the first successful jump. Due to fear of damaging the robot, we did not conduct enough trials to report a reliability metric for the box jumping controller on hardware.

4.6. Cassie Running Controller

Jumping motions have a significant, but singular, impact event. In contrast, running makes impacts with the environment at every stride. Because each stride leads immediately into the next, consistent tracking performance is essential for achieving stable running. Because bipedal running makes impact with its environment with only a single foot at a time, the space of states that are invariant is also larger for running. For these reasons, we develop a running controller as an additional application of the impact-invariant projection.

4.6.1. Controller Architecture

We track a SLIP-like pelvis trajectory and Raibert stepping generated footstep trajectories (Raibert et al., 1984) using the same OSC framework as used for the Cassie jumping controller. A diagram with the key elements is shown in Fig. 4.12.

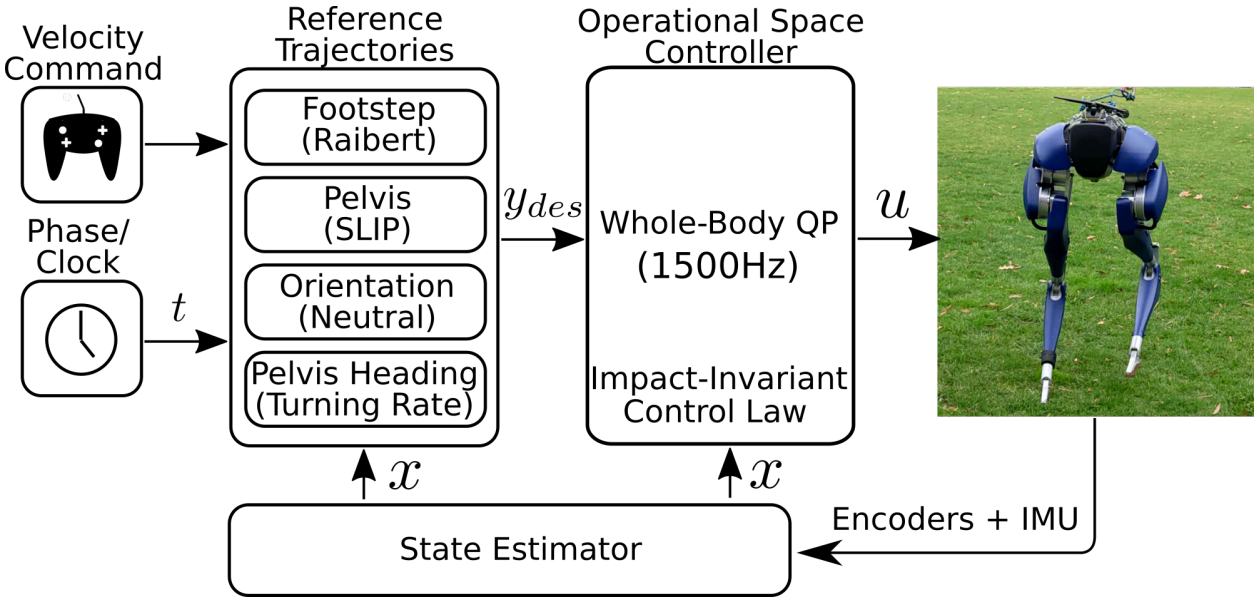


Figure 4.12: Key elements of the running controller diagram.

4.6.2. Reference Trajectory Generation

Mode timing

We use variable stance and flight durations to construct reference trajectories and determine contact mode switches. The nominal stance and flight durations, T_s and T_f respectively, are given in Table 4.3. However, we allow for variations in the timing to better align with the predicted touchdown and liftoff. The upcoming transition times for the full gait cycle are computed at each mode switch using the following heuristics:

$$T_s^* = \frac{l}{y_{SLIP}} T_s \tag{4.18}$$

$$T_f^* = \ddot{y}_{SLIP} + \sqrt{\dot{y}_{SLIP}^2 - 2g(l - y_{SLIP})}, \tag{4.19}$$

where we clip the modified stance and flight durations between $(1\pm\sigma_s)T_s$ and $(1\pm\sigma_f)T_f$ respectively to minimize timing changes in response to large disturbances.

The modified stance duration computes the ratio of the rest length to the SLIP length to roughly scale the liftoff time. The modified flight duration is computed using the time to touchdown assuming a ballistic trajectory.

SLIP-like pelvis trajectory

Inspired by the extensive literature on SLIP, we use a SLIP-like reference for the pelvis motion during stance {LS, RS}. We achieve a spring-like behavior by regulating the pelvis position relative to the current stance foot to a constant target height l and using the OSC gains to achieve the desired dynamics.

$$\ddot{y}_{SLIP,cmd} = K_p(l - y_{SLIP}) + K_d(\dot{y}_{SLIP}). \tag{4.20}$$

An important note is that we tune K_p and K_d to achieve the desired dynamics and not to achieve the best tracking. We adopt this simpler approach over tracking to true SLIP dynamics because the true dynamics are more difficult to regulate due to additional parameters and being purely an acceleration reference.

Footstep trajectories

Regulating the center of mass velocity is achieved through foot placement. While there are possible variations for choosing where to place the foot (Kajita et al., 2003) (Gong and Grizzle, 2022), the basic principle behind all the stepping strategies is stepping in the direction that you are falling. We choose to regulate the running velocity by planning footsteps with the widely recognized Raibert footstep control law (Raibert et al., 1984):

$$y_{ft,2} := \begin{bmatrix} y_{ft,x} \\ y_{ft,y} \\ y_{ft,z} \end{bmatrix} = \begin{bmatrix} \frac{v_x T_s}{2} + K_x(v_x - v_{des,x}) \\ \frac{v_y T_s}{2} + K_y(v_y - v_{des,y}) \\ -l \end{bmatrix}, \tag{4.21}$$

where K are the Raibert stepping gains, v_{des} are the desired velocities as commanded by the operator, and v is the current pelvis velocity computed by the state estimator. The x, y, and z subscripts in this context denote the sagittal, lateral, and vertical directions in the pelvis frame respectively. Finally, to avoid collisions between the legs, we offset the lateral target foot position by a constant distance c in the respective directions. $y_{ft,2}$ then defines the final target footstep location relative to the pelvis.

With the end footstep location $y_{ft,2}$ specified, we can generate a trajectory for the swing foot given its initial position $y_{ft,0}$ at liftoff to the final desired location. We specify all the reference trajectories as piecewise quadratic polynomials, so with the additional degrees of freedom we add a waypoint $y_{ft,1}$ so that the trajectory roughly resembles the swing leg retraction profile observed in both numerical optimization (Dai and Tedrake, 2012) (Seyfarth et al., 2003) and biology (Daley and Biewener, 2006). The full set of constraints defining the piecewise quadratic polynomial are as follows:

$$
\begin{aligned}
h &= [0, 0.6(T_s + 2T_f), (T_s + 2T_f)]^T \\
\sigma_0(h[0]) &= y_{ft,0} \\
\sigma_0(h[1]) &= y_{ft,1} \\
\sigma_1(h[1]) &= y_{ft,1} \\
\sigma_1(h[2]) &= y_{ft,2} \\
\dot{\sigma}_0(h[1]) &= \dot{\sigma}_1 h[1] \\
\ddot{\sigma}_0(h[1]) &= \ddot{\sigma}_1 h[1]
\end{aligned}
$$

where $y_{ft,0}$ is the initial foot position at liftoff, $y_{ft,2}$ is the desired footstep location at touchdown (4.21), and $y_{ft,1} = y_{ft,0} + 0.8(y_{ft,0} + y_{ft,2}) + [0, 0, d]^T$ is the waypoint we introduce to specify foot clearance. An illustration of the trajectory profile is shown in Fig. 4.13.

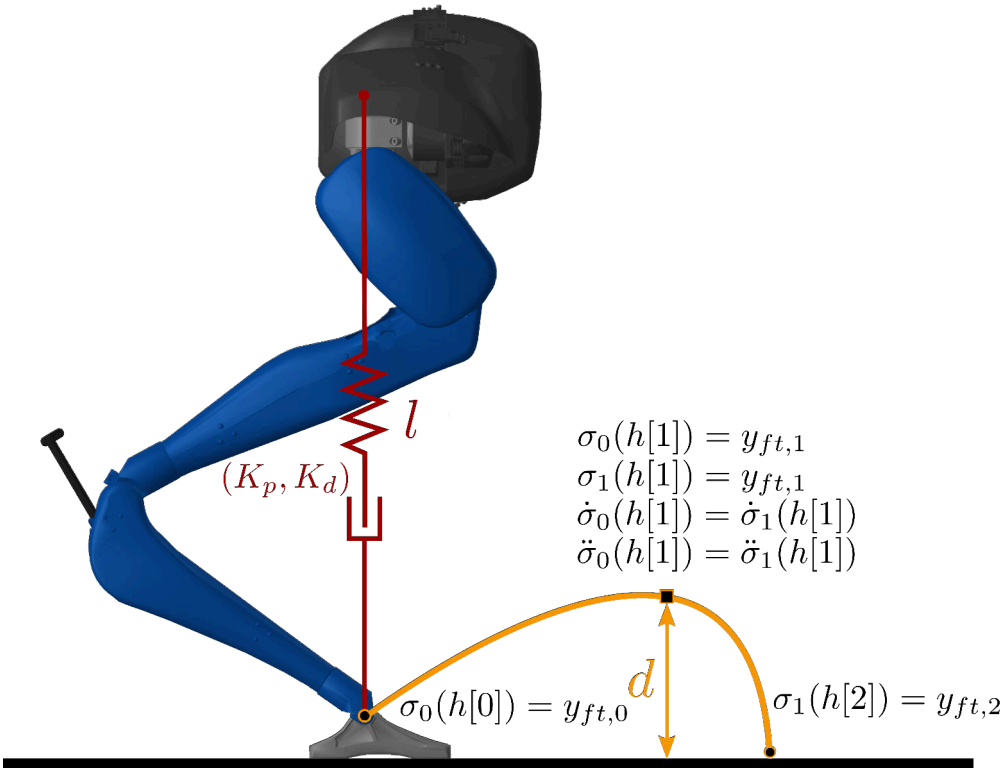


Figure 4.13: Illustration of key references for the running controller. The target height and swing foot trajectory with leg retraction profiles are shown in red and yellow respectively.

4.6.3. Reference Tracking

Finite State Machine

We use a time-based finite state machine (FSM) using the timings from Section 4.6.2 to specify the active contact mode and generate the clock signal for the reference trajectories. The set of finite states is {LS, LF, RS, RF}; L and R correspond to the left and right legs respectively and S and F correspond to Stance and Flight. We distinguish between the two air phases {LF, RF} to prescribe different tracking priorities for the different legs. The nominal durations for stance and flight deployed on hardware are reported in Table 4.3.

Tracking Objectives

The tracking objectives for the running controller are reported in Table 4.2. Although L_{SLIP} is defined as the position $\in \mathbb{R}^3$ of the pelvis relative to current stance foot expressed in the world

Table 4.2: Tracking objectives for the running controller. All values are expressed in the world frame.

Symbol	**Description**
$L_{SLIP} \in \mathbb{R}^3$	Pelvis position relative to the stance foot
$\psi \in SO(3)$	Pelvis Orientation
$y_L, y_R \in \mathbb{R}^3$	Foot position relative to pelvis
$\beta_L, \beta_R \in \mathbb{R}$	Hip yaw angle
$\phi_L, \phi_R \in \mathbb{R}$	Toe joint angle

Table 4.3: Relevant running controller parameters.

Parameter	**Symbol**	**Value**
Projection window	T	0.050 s
Blend time constant	τ	0.005 s
Stance duration	T_s	0.30 s
Flight duration	T_f	0.09 s
Pelvis target height	l	0.85 m
Foot clearance	d	0.2 m

frame, we only track the vertical component.

During left stance (LS), the active vector of tracking objectives is $[L_{SLIP}, y_R, \psi, \phi_R, \beta_R]$. For right stance (RS), the tracking objectives are the same, just mirrored for the other leg. During the aerial modes LF, RF, the active tracking objectives are $[y_L, y_R, \psi, \beta_L, \beta_R, \phi_L, \phi_R]$. The active tracking objectives per mode are also illustrated in Fig. 4.14.

Note, the dimension of the tracking objectives in flight, 13, is greater than the total degrees of actuation, 10. Therefore, during flight, the control formulation is overspecified and perfect tracking cannot be achieved. We choose to leave the problem overspecified as opposed to leaving out either the pelvis orientation or one of the foot targets because we found that trading off between multiple tracking objectives led to better performance on hardware.

Turning

We use the identity quaternion as the desired pelvis orientation and zero as the the desired angular velocity. By setting the task-space gains to $K_p = \text{diag}[150, 200, 0]^T$ and $K_d = \text{diag}[10, 10, 5]^T$, the

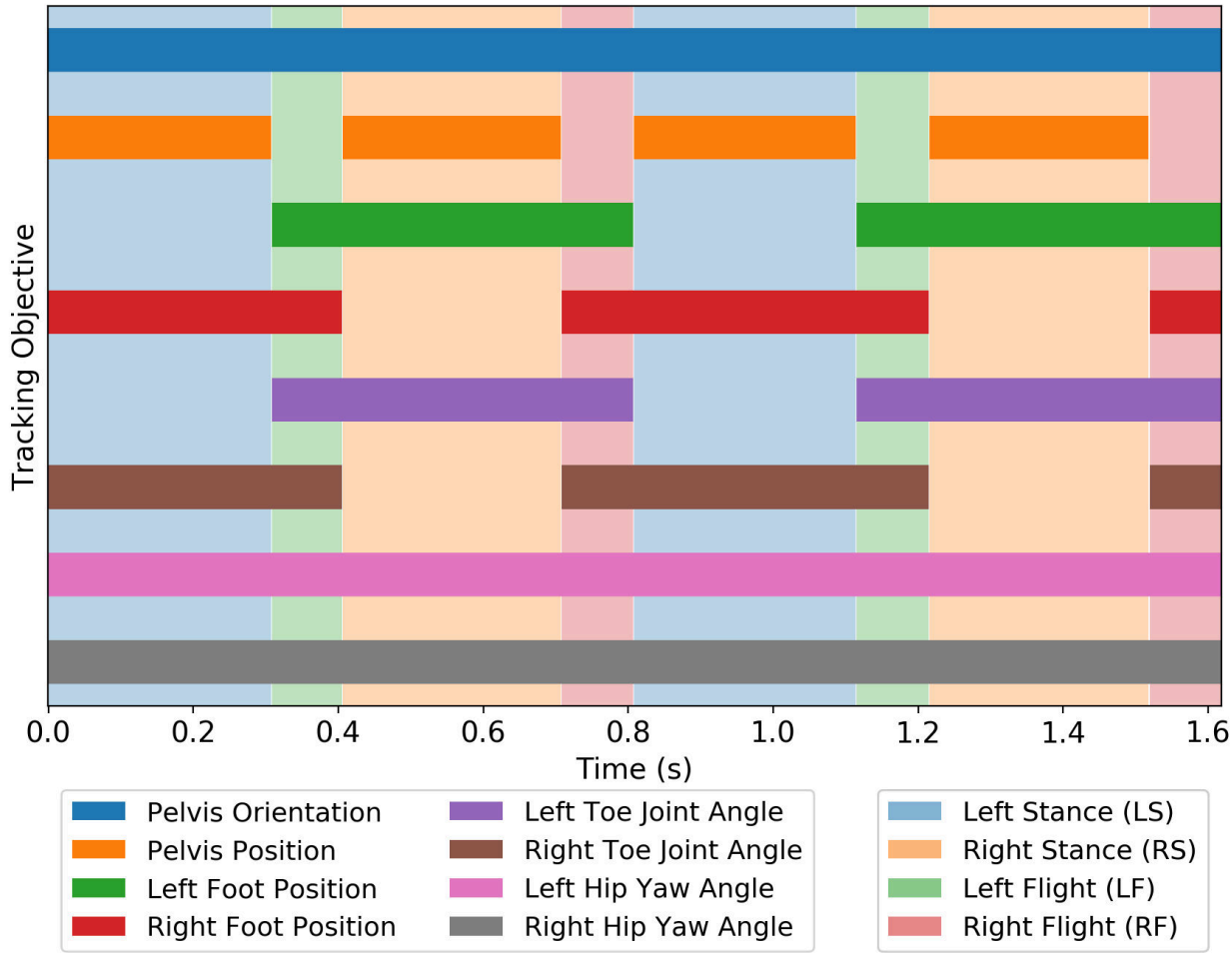


Figure 4.14: Active tracking objectives per mode for the running controller.

robot operator can specify the desired yaw velocity and achieve basic turning.

4.6.4. Simulation Results

The projection is an essential component for our framework to achieve stable running. Even in simulation, the default controller is immediately unstable when the foot touches the ground due the sensitivity to the impact event being coupled with the severe underactuation[6] of the running gait. For this reason, we only report comparisons with the no derivative feedback controller.

Both controllers are tasked with tracking a forward velocity command and the average errors over

[6]The running controller operates at the actuation limits of the knee motors and exhibits a significant flight phase, both which exacerbate the underactuation.

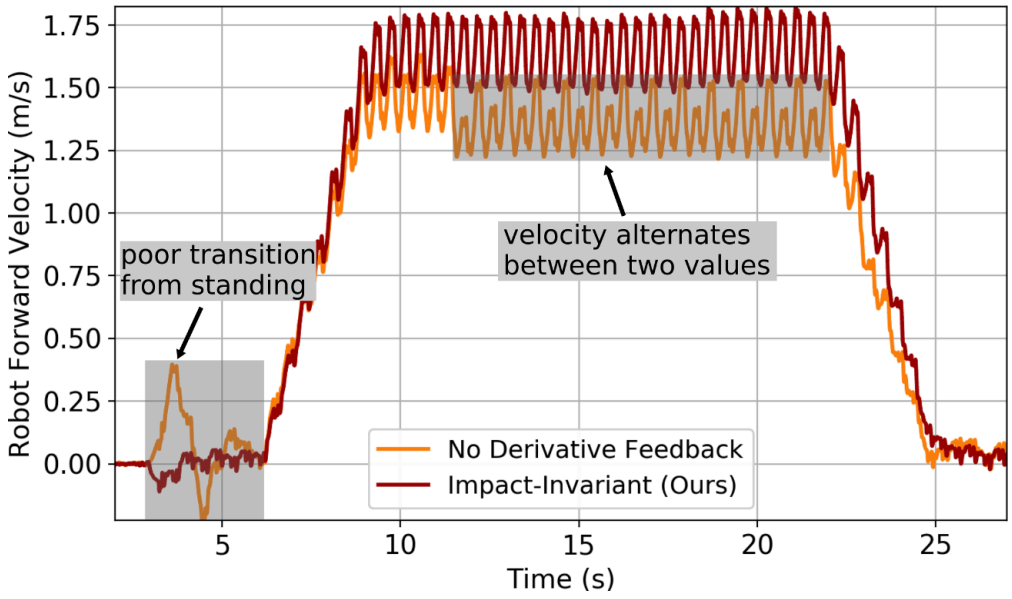


Figure 4.15: Comparison of our impact-invariant controller to the no-derivative feedback baseline for a simulated robot tracking a target velocity that is linearly ramped up and back down to 0. The velocity profile from the impact-invariant controller is noticeably smoother, which can be directly attributed to better tracking of objectives such as the foot trajectory. At higher velocities, the no-derivative feedback controller alternates between two velocities, which is undesirable. Comparisons to the default controller with no modifications are omitted because to the default controller is entirely unstable.

20 secs are reported in Table 4.4. The errors are computed as the position error at touchdown. The positions at touchdown are critical to stable running as they are the primary feedback mechanism for regulating the center of mass.

While the no-derivative controller has only slightly worse tracking performance on individual tracking objectives, the minor tracking discrepancies have a non-negligible effect on overall motion. As shown in Fig. 4.15, the no derivative controller is around 0.25 m/s slower than the impact-invariant controller with the same velocity command and has noticeably poorer tracking performance.

4.6.5. Hardware Results

We are able to achieve stable running, with the ability to command forward velocities and turn, on the physical robot using the same gains used in simulation. We tested the robot both indoors and outdoors on grass and on a turf field. The robustness of the impact-invariant control to early and late impacts on the physical robot is shown in Fig. 4.16. Videos of the experiments with the

Table 4.4: Average position tracking error compute at each foot touchdown of the impact-invariant controller and no derivative controller for the running controller tracking a constant forward velocity command. Results with the default controller are omitted because the default controller is entirely unstable for the running controller.

Objective	**Impact-Invariant Error**	**No Derivative Error**
Foot forward position	6.6 cm	6.6 cm
Foot lateral position	**3.6 cm**	4.4 cm
Foot vertical position	**3.9 cm**	5.6 cm
Hip yaw angle	**0.026 rad**	0.034 rad
Toe joint angle	0.030 rad	**0.020 rad**

physical robot are included in the supplementary video. The most relevant controller parameters of the running controller are provided in Table 4.3, and the full running parameters are reported in Table 4.8.

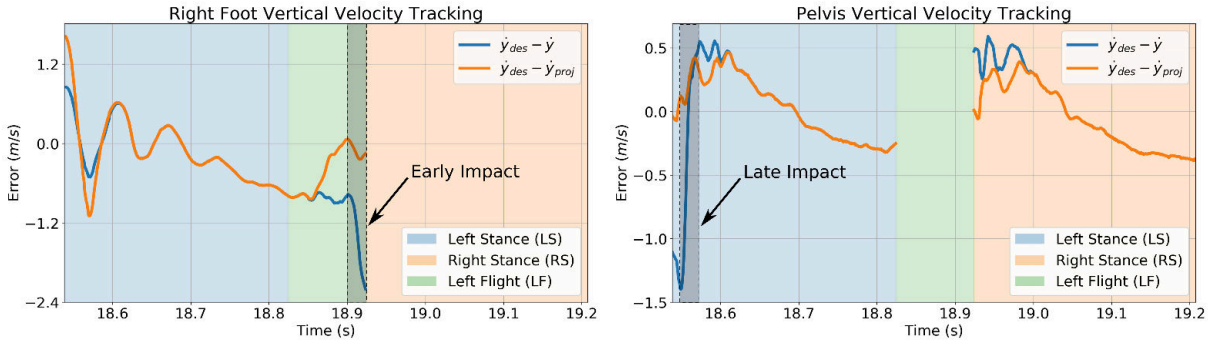


Figure 4.16: Velocity tracking errors from the running controller on the physical Cassie robot. During the experiment, the robot makes impact *late* with its left foot, which initially results in a large tracking error for the pelvis target. The robot also make impact *early* with its right foot, which results in a large tracking error for the right foot target. In both cases, the impact-invariant projection protects the controller from overreacting to the mismatch in impact timing.

4.7. Discussion

In this paper, we introduce a general strategy that enables controllers to be robust to uncertainties in the impact event without sacrificing control authority over unaffected dimensions. The strategy makes use of an easy-to-compute modification of how the velocities of the robot enter the controller. Specifically, we project the velocity error into a subspace that is invariant to the impact event. This projection completely eliminates sensitivity of the controller to potential contact impulses, while

minimally deviating from the original controller.

We demonstrate through examples with legged robots that the impact-invariant projection is robust to the impact event, while achieving better tracking performance compared to alternative methods. The modification can easily be applied to controllers for complex bipeds such as Cassie, and the modification enables highly dynamic motions such as jumping and running.

4.7.1. Recovered Control Authority

We only apply the projection in a small time window around an impact. An alternative to the projection is to instead turn off all derivative feedback as is done during the contact transition mode van Steen et al. (2022). If impacts are infrequent and short, the additional benefit gained from using the impact-invariant projection is minimal. However, for motions where impacts are frequent such as running, the additional benefit can be substantial. Our running controller uses a projection window of 50 ms on both sides of the impact event for a total of 100 ms when the projection is active. The nominal stepping period of the running controller is 0.39 s, so the projection is active for over 25% of the time. Even tighter window durations cover a non-negligible duration of the entire motion.

4.7.2. Introducing Additional Controller Invariances

The impact-invariant framework enables robustness to a very particular source of uncertainty: impacts. It seems straightforward to extend this idea to eliminate sources of uncertainty from the control law, but on further examination impacts may be a very specific case of where this method would be useful. The key observation of this work is that impacts are *brief* periods of *high uncertainty* that enter the dynamics in a highly structured manner. Therefore, a reasonable approach is to effectively ignore the uncertainty in the short amount of time when the magnitude of the uncertainty is at its greatest. Other sources of uncertainty such as model differences or uncontrollable elements such as physical springs do not have these attributes. We cannot ignore model differences as they permeate the entire dynamics and are persistent throughout, and while springs only enter the dynamics at a single joint, their oscillations do not resolve in a short amount of time.

4.7.3. Future Work

Although the examples featured in this work focus on legged locomotion, the method is general and can be applied to other rigid body systems with impacts, such as manipulation. For future work, we plan to investigate how this method can be applied to high-speed grasping. A complementary avenue of future research lies in how we can leverage other sensor modalities such as tactile sensing as a method to have a less conservative bound on impact uncertainty by working with partial sensor feedback.

Reinforcement learning has produced controllers with similar dynamic motions on Cassie Siekmann et al. (2021a) Li et al. (2024) with impressive robustness. Another avenue of future research is to evaluate whether these learned controllers exhibit similar invariances and whether incorporating such invariances into the learned controllers might improve robustness.

4.8. Appendices

Table 4.5: Length, mass and inertia parameters for the planar five-link biped evaluated in simulation. Values are taken from the physical robot, Rabbit Chevallereau et al. (2003).

	thigh	shin	torso
Length (m)	0.4	0.4	0.625
Mass (kg)	6.8	3.2	12.0
Inertia along principal axes (kgm^2)	0.47	0.20	1.33

Table 4.6: Full impact-invariant parameters and regularization weights for the Cassie jumping controller.

Symbol	**Description**	**Value**
μ	Friction coefficient	0.6
T	Projection window	0.05 s
τ	Blend time constant	0.005 s
W_u	Regularization weight on inputs	1e-6
W_{acc}	Regularization weight on generalized accelerations $\ddot{q}$	1e-4
W_λ	Regularization weight on contact forces	1e-1

Table 4.7: Feedback gains for the Cassie jumping controller (jump and box jump) deployed on hardware. Weight and gain matrices are diagonal matrices, represented here as vectors to be concise.

OSC Objective	W	K_p	K_d
Toe joint angle	0.01	1500	10
Hip yaw angle	2.5	100	5
Pelvis [x, y, z]	$[20, 2, 20]$	$[40, 50, 40]$	$[7.5, 5, 5]$
Pelvis [roll, pitch, yaw]	$[10, 5, 1]$	$[150, 200, 150]$	$[10, 10, 5]$
Foot [x, y, z]	$[10, 100, 10]$	$[125, 50, 150]$	$[2.5, 2.5, 0]$

4.8.1. Least Squares and Projections

It is well studied that the least squares problem is equivalent to a projection Strang (2022). We show here that our choice of least squares in 4.4 is equivalent to the impact-invariant projection matrix in 4.3 when the tracking objectives are the generalized robot states, $y = q$, $\dot{y} = \dot{q}$, and there are no holonomic constraints $J_h = 0$. Recall, $J_y = \frac{\partial y}{\partial q}$, so in this case $J_y = \frac{\partial q}{\partial q} = I$, where I is the identity matrix. We can simplify 4.6 using these assumptions to arrive at

$$\begin{bmatrix} \lambda^* \end{bmatrix} = \begin{bmatrix} A^T A \end{bmatrix}^{-1} \begin{bmatrix} A^T (\dot{q}_{des} - \dot{q}). \end{bmatrix},$$

where $A = M^{-1} J_\lambda^T$. We can substitute λ^* into 4.7

$$\dot{q}_{proj} = \dot{q} + A(A^T A)^{-1} A^T (\dot{q}_{des} - \dot{q})$$

to arrive at the projected generalized velocities. $\dot{y}_{proj} = \dot{q}_{proj}$ so we can express the projected velocity tracking error $\dot{\tilde{y}}_{proj}$ as:

$$\begin{aligned} \dot{\tilde{y}}_{proj} &= \dot{q}_{des} - (\dot{q} + A(A^T A)^{-1} A^T (\dot{q}_{des} - \dot{q})) \\ &= (I - A(A^T A)^{-1} A^T)(\dot{q}_{des} - \dot{q}) \\ &= Q(\dot{q}_{des} - \dot{q}), \end{aligned}$$

which is identical to the result from the impact-invariant projection.

Table 4.8: Full trajectory, impact-invariant parameters, and regularization weights for the Cassie running controller deployed on hardware.

Symbol	Description	Value
μ	Friction coefficient	0.6
T	Projection window	0.05 s
τ	Blend time constant	0.005 s
l	Pelvis target height	0.85 m
T_s	Stance duration	0.3 s
T_f	Flight duration	0.09 s
σ_s	Stance duration variance	0.2
σ_f	Flight duration variance	0.1
c	Footstep lateral offset	0.04 m
d	Foot clearance	0.2 m
K_x	Raibert footstep sagittal feedback	0.01
K_y	Raibert footstep lateral feedback	0.3
W_u	Regularization weight on inputs	1e-6
W_{acc}	Regularization weight on generalized accelerations $\ddot{q}$	1e-4
W_λ	Regularization weight on contact forces	1e-1

4.8.2. Hardware Setup

All processing is done on the Intel NUC 11 computer onboard Cassie. Note we swapped the original Intel NUC onboard Cassie to take advantage of the faster compute of a newer CPU. We run the state estimator and the controllers asynchronously as separate processes, and communication between processes is handled using LCM Huang et al. (2010), while user commands are sent through the radio remote.

State Estimator

We use the contact-aided invariant EKF developed in Hartley et al. (2020) to estimate the floating-base pelvis state. Although we do not directly use contact detection in our controllers, the state estimator utilizes the current contact mode estimate in the measurement update. We achieve this using a generalized-momentum observer, similar to the method used in Bledt et al. (2018b), to estimate the contact force at each foot. We then set a threshold of 60 Nm on the contact normal force to define contact. We observe that this has a faster response and better accuracy over detecting

Table 4.9: Feedback gains for the Cassie running controller deployed on hardware. Weight and gain matrices are diagonal matrices, represented here as vectors to be concise.

OSC Objective	W	K_p	K_d
Toe joint angle	0.01	1500	10
Hip yaw angle	2.5	100	5
Pelvis [x, y, z]	$[0, 0, 5]$	$[0, 0, 115]$	$[0, 0, 5]$
Pelvis [roll, pitch, yaw]	$[10, 5, 1]$	$[150, 200, 0]$	$[10, 10, 5]$
Foot [x, y, z]	$[10, 100, 10]$	$[125, 75, 75]$	$[5, 5, 5]$

contact using spring deflections. The state estimator runs at 2000 Hz.

Controller Implementation

We write our controllers using Drake Tedrake and the Drake Development Team (2019) for the systems framework and all rigid body kinematics calculations. To set different tracking priorities during the flight phase for the running controller, we linearly ramp the weight for the foot tracking objective from 0.5 to 4 times the specified value across the duration of the trajectory. Although the tracking objectives are expressed in the world frame, we compute the errors in the robot yaw frame in order to have the gains operate on the sagittal and lateral directions rather than the world x and y directions. Finally, we solve the OSC QP using a minor modification of the OSQP Stellato et al. (2020) interface provided by Tedrake and the Drake Development Team (2019) at 1500 Hz.

CHAPTER 5

Controlled Sliding

Parts of this chapter were previously published as parts of William Yang and Micahel Posa. Dynamic On-Palm Manipulation via Controlled Sliding. In *Robotics: Science and Systems (RSS)*, Delft, Netherlands, 2024. URL https://roboticsconference.org/program/papers/12/

In this chapter, we move away from legged robots and instead turn our attention to dexterous dynamic manipulation. Here we introduce two challenging dynamic manipulation tasks that utilize the full set of contact modes: sticking, sliding, and no contact. We accomplish these tasks using a framework that leverages recent advancements in contact-implicit MPC. Despite the complexity of the tasks, we do not rely on common aids such as reference trajectories or motion primitives, displaying promising generality. The work in this chapter has an accompanying project website: https://dynamic-controlled-sliding.github.io/.

This chapter is ordered as follows. Section 5.1 provides motivation. Section 5.2 details related work. Section 5.3 introduces the problem setup, while Section 5.4 details how we decompose the system into two separate models. We define the control framework in Section 5.5 and provide experiment details in Section 5.6. Finally, we discuss results and limitations in Section 5.7 and Section 5.8 respectively.

5.1. Introduction

Recent advancements in robot manipulation have demonstrated impressive dexterity (Chen et al., 2023) and generality (Curtis et al., 2022) (Chi et al., 2023). However, these methods largely focus on slow tasks that can be viewed from a quasi-static perspective. As robots are increasingly being asked to perform manipulation tasks in logistics applications such as warehouse robotics, speed becomes a key driving metric. While there are plenty of examples of dynamic manipulation, the methods are often achieved using ad-hoc, task-specific solutions (Ruggiero et al., 2018) and demonstrated on systems with few degrees of freedom and few contacts. The desire for a general control

framework for contact-rich tasks has resulted in many formulations for contact-implicit model predictive control (Aydinoglu et al., 2024) (Kurtz et al., 2023) (Le Cleac'h et al., 2024) (MPC), which can automatically plan when and where to make and break contact and are reported to be fast enough for real-time control.

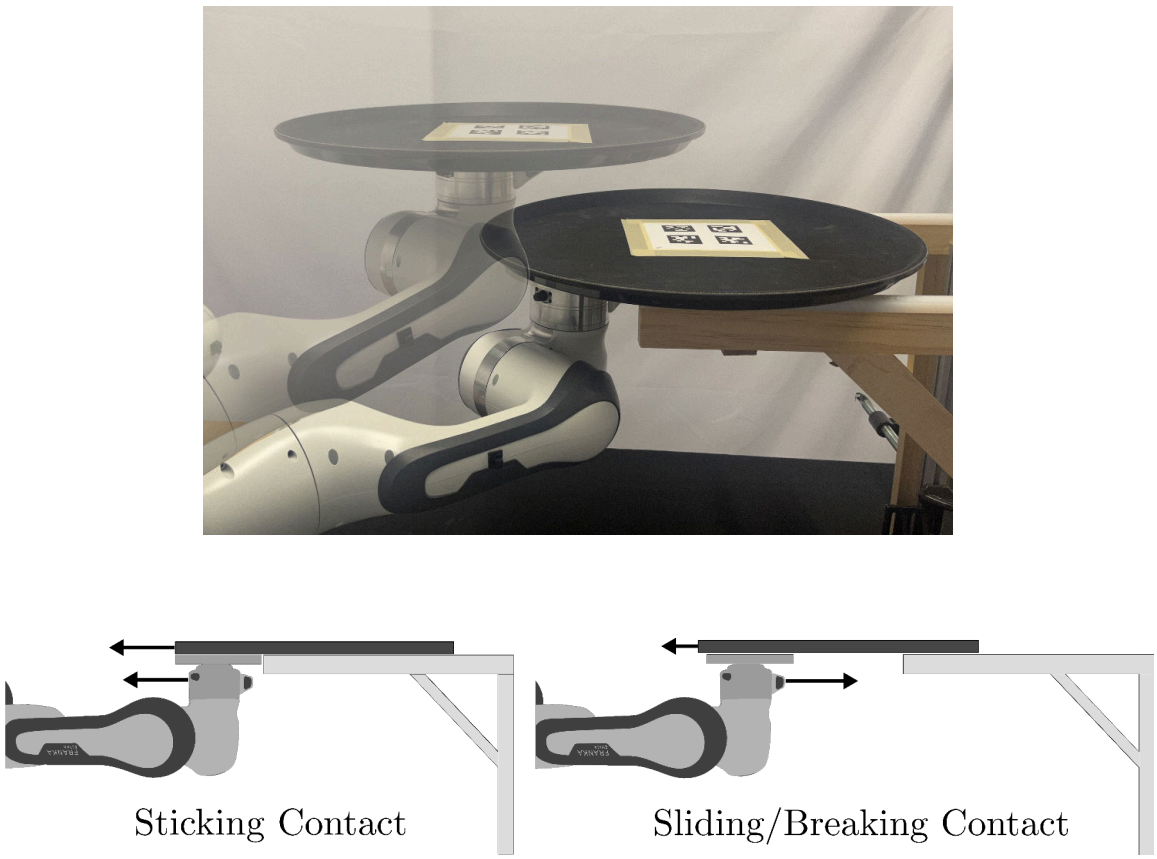


Figure 5.1: We examine a dynamic sliding task, where the robot uses the full spectrum of contact modes (sticking, sliding, making and breaking contact) in order to retrieve a tray resting on external supports. We use contact-implicit MPC to automatically plan when and where to use different contact modes. With careful consideration on how to integrate the simplified MPC model with the robot arm, we are able complete the entire maneuver of retrieving the tray, lifting it, and placing it back on the external supports in just 5 seconds, demonstrating dynamic capability for a contact-rich task.

In this chapter, we focus on an extension of the waiter's task to serve as an example of a general class of problems that involve multiple contacts, reasoning about external contact, as well as stick-slip behavior. This task both resembles real dexterous manipulation skills and also exemplifies a range of challenges faced in general-purpose dexterous manipulation. In contrast with prior renditions of the

waiter's task (Subburaman et al., 2023) (Heins and Schoellig, 2023) (Brei et al., 2024), which focus on just transporting the tray while maintaining static contact, our task simulates the full process of first retrieving the tray, then lifting the tray, and finally placing the tray back at its initial position. The tray initially rests on external supports, so that a portion of the tray hangs over the edges of the supports. As illustrated in 5.1, in order to retrieve the tray, the robot must first shift the tray so that it slides onto the end effector before it can be supported from underneath. Similarly, in order to place the tray back at its initial position, the end effector must shift the tray forward onto the supports. Both of these maneuvers require repeated stick-slip transitions. The primary challenge of this task is the consideration of dynamic frictional contact. One major challenge of frictional contact is the known model inaccuracies of Coulomb friction and rigid frictional impacts (Remy, 2017). Another major challenge is that sliding adds additional contact modes to the already challenging hybrid planning problem. Prior works that consider control with sliding contact are restricted to a single contact for planar dynamic tasks (Shi et al., 2017) (Hou et al., 2020) or multiple contacts for planar quasi-static tasks (Doshi et al., 2022) (Taylor et al., 2023). Other methods can reason about sliding contacts in 3D quasi-static and quasi-dynamic tasks (Cheng et al., 2022) (Cheng et al., 2023); however, the methods are currently too slow for real-time control.

Surprisingly, we show that, with some improvements, a general contact-implicit model predictive controller (MPC) framework can accomplish this dynamic task. Specifically, we build upon prior work by carefully considering the integration between the simple model used by the MPC and the low-level tracking controller in order to accurately track the dynamic motions commanded by the MPC. Our controller automatically plans motions with repeated stick-slip transitions as the robot pushes or pulls the object, including initiating slipping to reposition the end effector to then push or pull again. The controller accomplishes this all without using heuristics or commonly relied on aids such as reference trajectories or motion primitives.

5.2. Related Work

5.2.1. Contact Mode Regulation

The primary exploration of this work is how to plan and regulate between frictional contact modes (sliding, sticking, and breaking contact altogether), with an emphasis on dynamic sliding contact as it is comparatively unexplored. Dynamic sliding (Shi et al., 2017) and pivoting (Hou et al., 2020) for object reorienting and regrasping are performed by simultaneously regulating inertial and frictional forces using a parallel jaw gripper. However, these works are limited to the planar case and only consider a single surface contact. Doshi et al. (2022) demonstrate impressive control of both sliding and sticking contact along multiple surfaces including utilizing external contacts (Taylor et al., 2023); however, they focus on quasi-static manipulation and again are limited to a planar system. Woodruff and Lynch (2017) demonstrate sequencing multiple motion primitives (Lynch and Mason, 1999), including dynamic sliding, demonstrated on a planar manipulator and block set up. However, the full trajectory is planned offline and relies on local feedback control to stabilize each motion primitive.

We highlight the planar nature of prior examples because planning for sliding contact in 3D is fundamentally more challenging than in the planar case. This is because, in addition to the increased state dimension, the planar case only requires consideration of 4 hybrid modes (sticking, no contact, slide left, slide right) per contact, whereas there exists a *continuum* of sliding modes for the 3D case. Higashimori et al. (2009) considers full surface-surface friction when manipulating a flat object with 3 degrees of freedom (DOFs) resting on a pizza peel-like platform with only controlled 2 DOFs. Their work showcases impressive controllability, but the method assumes that the platform is much larger than the object and the nominal pressure distribution on the object being uniform.

5.2.2. Waiter Task

Several works (Pham et al., 2017) (Heins and Schoellig, 2023) (Subburaman et al., 2023) (Brei et al., 2024) have tackled the “waiter task", where objects are balanced on top of an end effector with a planar surface. Despite the similar task set-up, all of these works critically focus on avoiding sliding between the object and manipulator, whereas the key focus of our work is to specifically leverage

sliding to perform tasks that would otherwise be infeasible.

5.2.3. Contact-Implicit MPC

Recently, several contact-implicit MPC frameworks have demonstrated solve times fast enough for real-time control on systems with multiple contacts and many degrees of freedom (Aydinoglu et al., 2024) (Kurtz et al., 2023) (Le Cleac'h et al., 2024). However, these methods have not been evaluated on dynamic manipulation with sliding contacts. Here, we do not propose a new MPC framework, but rather we seek to identify the implementation details to adapt such a framework to a dynamic task, including how to consume the outputs in a downstream tracking controller. Critically, the output of these contact-implicit methods were tracked as position set-points stabilized with impedance gains (Aydinoglu et al., 2024) (Kurtz et al., 2023), whereas we track time-parameterized trajectories for the end effector position and end effector forces. Position set-points rely on the stiffness of the impedance controller to achieve accelerations, while accelerations and forces can be specified directly and are defined smoothly using time-parameterized trajectories.

5.3. Problem Setup

We are interested in the problem setup shown in Fig. 5.2. The system consists of a serial-link manipulator equipped with a small flat disk as its end effector, where the end effector is constrained to move only in 3D translation. The robot arm is tasked with retrieving, lifting, and returning (placing) a tray as shown in Fig. 5.2. The tray is initially resting on external supports and starts in a slightly overhanging position so that it can be contacted on its bottom surface. The tray has all floating base degrees of freedom, with its pose in $SE(3)$, and its pose can be tracked using fiducials attached to the tray. We assume that we have accurate model parameters (mass, inertia, geometry, and friction) of each component (robot arm, end effector, tray, and supports), although we do examine the effect of inaccurate models in Section 5.7. We assume a single coefficient of friction per pair of geometries (tray/end effector and tray/supports). The task objectives: retrieve, lift, and place, are specified as three sequential targets, meaning the next target is given when the tray reaches the previous target.

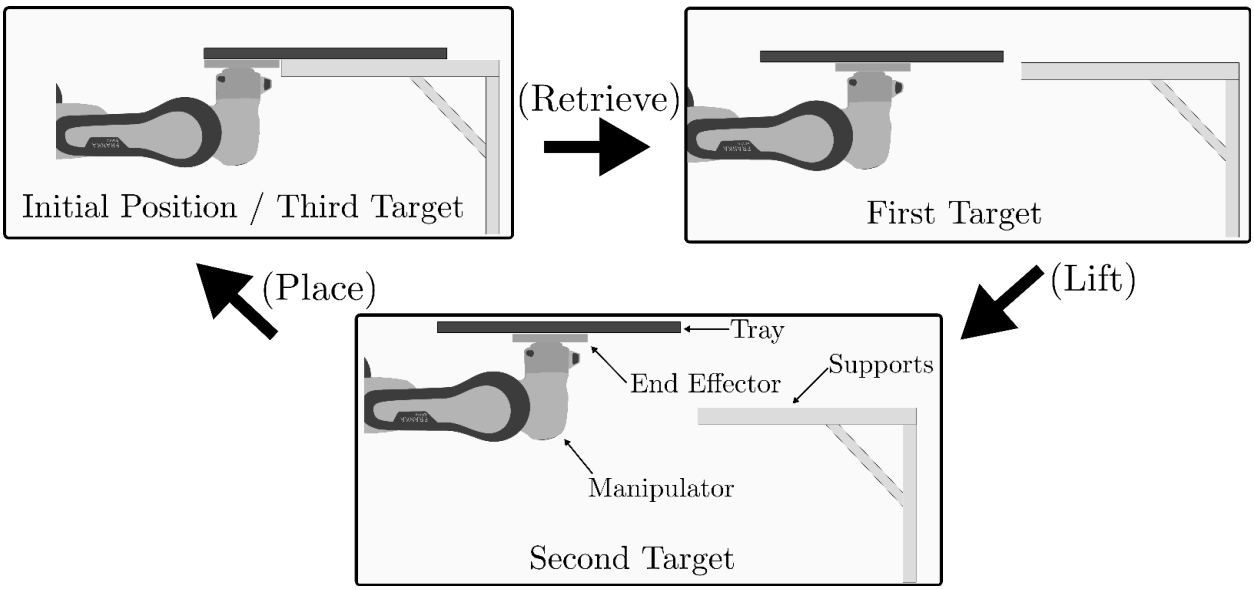


Figure 5.2: The three target positions. The grasp locations on the tray change between targets, thus requiring the end effector to either slide and/or break contact with the tray.

5.4. System Models

In this paper, we abstract the system using two models as shown in Fig. 5.3. We model the the end effector, object, and external contacts as a Linear Complementarity System Heemels et al. (2000) (LCS) to use for the MPC. For the low-level operational space controller, we only consider the dynamics of the robot arm and rely on inputs from the MPC to address the interaction forces from the object.

5.4.1. Linear Complementarity Model

We model the dynamics of the end effector, object, and external contacts as a discrete time LCS. We ignore rest of the robot arm in the MPC model by considering the end effector as an isolated floating object with only translation degrees of freedom and controlled directly with forces applied to its center of mass. To ensure downstream feasibility when applying this model to the actual system, we impose workspace and input limits on the MPC model. The state of the LCS $x_{lcs} = [q_{lcs}, v_{lcs}]$ is a combination of the positions $q_{lcs} = [q_{ee}, q_{obj}]$ of the end effector and object and the corresponding velocities $v_{lcs} = [v_{ee}, v_{obj}]$.

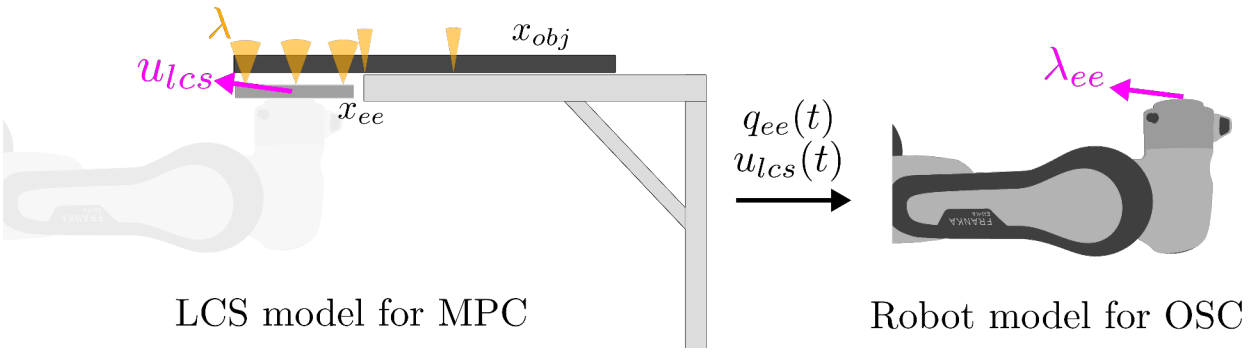


Figure 5.3: We abstract the system into two models. The LCS model captures the contact forces λ between the end effector, tray, and supports. In the LCS model, the robot arm is abstracted away and replaced with direct inputs to the end effector u_{lcs}. We then use a robot-only model to track the end effector position $q_{ee}(t)$ and force $u_{lcs}(t)$ trajectories commanded from the MPC, so $\lambda_{ee} = u_{lcs}$.

The full LCS state vector x_{lcs} is therefore

$$q_{lcs} = \begin{bmatrix} ee_x \\ ee_y \\ ee_z \\ tray_{qw} \\ tray_{qx} \\ tray_{qy} \\ tray_{qz} \\ tray_x \\ tray_y \\ tray_z \end{bmatrix}, v_{lcs} = \begin{bmatrix} ee_{vx} \\ ee_{vy} \\ ee_{vz} \\ tray_{wx} \\ tray_{wy} \\ tray_{wz} \\ tray_{vx} \\ tray_{vy} \\ tray_{vz} \end{bmatrix},$$

The notation is as follows, $()_x$ indicates the x position, $()_{qw,qx,qy,qz}$ is the orientation expressed as a quaternion, $()_{vy}$ indicates the y velocity, and $()_{wz}$ expressed the angular velocity. All quantities are expressed in the world frame. The control input $u_{lcs} = [u_x, u_y, u_z]$ to the LCS are forces directly applied to the end effector, such that it can be controlled in 3D translation.

The dynamics of the LCS have the form:

$$x_{k+1} = Ax_k + Bu_k + D\lambda_k + d \tag{5.1}$$

$$0 \leq \lambda_k \perp Ex_k + F\lambda_k + Hu_k + c \geq 0, \tag{5.2}$$

where $x_k \in \mathbb{R}^{n_x}$, $\lambda_k \in \mathbb{R}^{n_\lambda}$, and $u_k \in \mathbb{R}^{n_u}$ are the state, force, and input variables at the k-th knot point. (5.1) are the system dynamics linearized at the current state and input. The $\perp$ indicates a complementarity constraint, where $0 \leq \lambda \perp \phi \geq 0$ implies $\lambda \geq 0$, $\phi \geq 0$, $\lambda^T \phi = 0$. Critically, this constraint succinctly describes the multi-modality of contact dynamics for both making and breaking contact as well as the boundary between stick and slip. For example, the boundary between sliding and sticking friction for a given sliding direction can be expressed as:

$$0 \leq \mu\lambda_n - \lambda_t \perp v \geq 0, \tag{5.3}$$

where λ_n is the normal force, and λ_t is the tangential force in the opposite direction of the sliding velocity v. With this context, (5.2) is the linearization of the contact boundaries at the current state and input.

The contact dynamics of the LCS are governed by our choice of contact geometry. We approximate the surface-surface contacts between the end effector and object as well as the object and external contacts using point contacts. We use three contact points between the end effector and object, because that is the minimum number necessary to have statically stable surface-surface contact. Similarly, we model each support as two points to represent the line contacts. These contact geometries are visualized in Fig. 5.4. Under this choice of contact geometry, ϕ encodes the distance between any of these contact points and the tray, represented as a cylinder. Under this modeling choice, the number of contacts n_λ is fixed. Note, we are ignoring potential contacts between the end effector and the supports. As these contacts are undesirable, we simply avoid these contacts by imposing workspace constraints on the end effector.

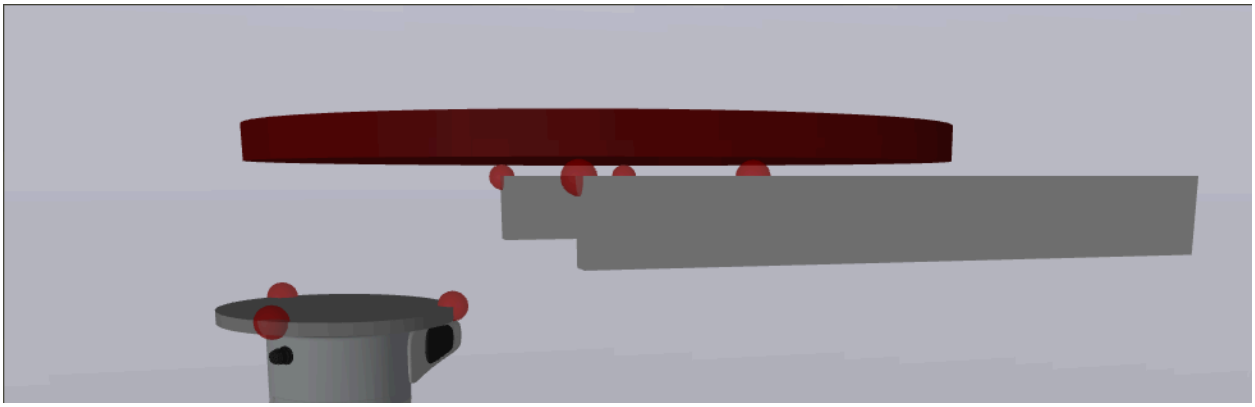


Figure 5.4: We consider seven total contacts for our task. The contact geometries shown in red. We represent the tray as a cylinder and we choose fixed contact points on the end effector and supports, which we model as spheres. The radii for the spheres are enlarged by a factor of 10 for visibility purposes. A minimum of three contact points are required to approximate surface-surface contact between the end effector and tray, while two contact points are required to model each line contact from the supports.

5.4.2. Robot-Only Model

We only consider the state of the arm when applying our low level tracking controller. We denote the state of the robot arm as $x_{arm} = [q_{arm}, \dot{q}_{arm}]$, which is comprised of its generalized positions $q_{arm} \in \mathbb{R}^{n_{arm}}$ and generalized velocities $\dot{q}_{arm} \in \mathbb{R}^{n_{arm}}$. The arm is controlled using actuator inputs $u_{arm} \in \mathbb{R}^{n_{arm}}$, where the inputs are motor torque commands. For brevity, we omit the $_{arm}$ subscript for the remainder of this section. We can describe the arm dynamics using the manipulator equation:

$$M(q)\ddot{q} + C(q, \dot{q}) = Bu + J(q)^T \lambda_{ee}, \tag{5.4}$$

where M is the mass matrix with approximated reflected inertia terms Featherstone (2014), C contains the Coriolis and gravitational forces, B maps actuator inputs to generalized forces, and J is the contact Jacobian that maps forces λ_{ee} applied at the end effector to generalized forces.

5.5. Methods

5.5.1. Complementary Consensus Control

We formulate our control problem as a contact-implicit MPC problem with LCS dynamics. This is succinctly formulated as the following optimization problem:

$$\min_{x_k, u_k, \lambda_k} \quad x_N^T Q_f x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k \tag{5.5}$$

$$s.t. \quad x_{k+1} = Ax_k + D\lambda_k + Bu_k + d \tag{5.6}$$

$$Ex_k + F\lambda_k + Hu_k + c \geq 0 \tag{5.7}$$

$$\lambda_k \geq 0 \tag{5.8}$$

$$\lambda_k^T (Ex_k + F\lambda_k + Hu_k + c) = 0 \tag{5.9}$$

$$x_{min} \leq x_k \leq x_{max} \tag{5.10}$$

$$u_{min} \leq u_k \leq u_{max}, \tag{5.11}$$

where N is the planning horizon, Q_f, Q, R are cost matrices, and $x_{min}, x_{max}, u_{min}, u_{max}$ are bounds on the state and input variables.

(5.6), (5.7), and (5.8) are the dynamics constraints of the LCS. (5.9) is the orthogonality constraint for the complementarity. Note, (5.9) is non-convex, but it is possible to introduce binary variables to represent the contact modes and transcribe the entire problem as a Mixed Integer Quadratic Program (MIQP). However, this scales poorly with the number of contacts, as a binary variable is needed for each contact across all knot points.

Instead, we adopt a method called Complementarity Consensus Control (C3) Aydinoglu et al. (2024), which addresses the scaling problem by decoupling the time dependence of the contact decisions. The algorithm is based in consensus alternating direction method of multipliers (ADMM), which optimizes over multiple copies of the decision variables.

The full details of the algorithm is outside of the scope of this paper, but a key property is that

the algorithm alternates between solving the MPC problem as a quadratic program (QP) without complementarity constraints and projecting the current MPC solution to the complementarity constraints as a mixed integer quadratic program (MIQP) *separately* for each knot point. While the solutions will eventually converge to each other, we choose to terminate early after a fixed number of iterations on a potentially suboptimal solution. We choose to terminate after the QP step, because we empirically observe better performance. Note, the suboptimal solutions from terminating early do not necessarily satisfy the full LCS dynamics, but in practice are good enough when used in MPC for even contact-rich tasks.

5.5.2. System Linearization

We approximate our system as a LCS at each C3 solve. The continuous dynamics parameters (A, B, d) of the LCS can be solved via automatic differentiation using any popular rigid body dynamics library. The gap function ϕ and corresponding contact Jacobians J for convex geometries can be computed by a library that implements collision detection, e.g. via the GJK algorithm(Gilbert et al., 1988). With ϕ and J for each contact along with a choice of force basis, we can compute the contact-related terms: D, E, F, H, c. We use the convex time-stepping contact model proposed by Anitescu and Potra (1997) to form the force basis. In this model, contact forces are parameterized via the extreme rays of the pyramidal approximation of the friction cone. That is, for a square pyramidal approximation, there are 4 contact force variables per point contact. This choice of contact force basis is visualized in Fig. 5.8.

5.5.3. MPC Modifications for Dynamic Motions

The fast motions commanded by our MPC are on the same timescale as the solve time. For this reason, the system state at the end of the MPC solve is likely far from the system state at the beginning of the MPC solve. We address this latency problem by using the predicted state of the system according to the previous MPC solve as the initial state constraint similar to (Bledt, 2020)

$$x_0 = x_{sol}(\overline{dt}), \tag{5.12}$$

where $x_{sol}(t)$ is the state trajectory from the previous MPC solve and $\overline{dt}$ is the filtered average MPC solve time. Because we have less confidence in the accuracy of our contact models, we only apply this prediction to the end effector state and not the state of the tray. Note, by using the predicted state we are eliminating the MPC feedback capabilities on the end effector. To avoid the predicted state from significantly deviating from the measured state, we saturate the difference between the predicted state and the measured state. In practice, this has a minimal effect as because the low-level tracking controller for the end effector enforces that the true state matches the predicted state.

Warm starting by giving the MPC an initial guess from the previous solve is a common technique to reduce computation time. However, the values from the previous solution are often poor initial guesses because the contact modes at each knot point planned from the current MPC state may differ from the previous solution. Because the dynamics can vary greatly between contact modes, so can the values for x, u, λ. For this reason, we use the corresponding predicted values from the previous solution when possible for warm starting. Additionally, C3 involves multiple QP and MIQP solves per MPC solve each with different cost parameters. We address this by treating each QP and MIQP as separate optimization programs each with a separately cached set of warm start variables in order to increase the quality of the warm start.

5.5.4. Operational Space Control

We use the operational space controller (OSC) introduced in Section 3.3.2 to stabilize the plans commanded by the MPC. Specifically we track the end effector position, orientation, and force applied at the end effector specified as time-parameterized trajectories.

We formulate this as the following quadratic program (QP), simplified from the more general formulation presented in Section 3.3.2:

$$\min_{u,\lambda,\ddot{q}} \quad \|\lambda - \lambda_{ee}\|_W^2 + \sum_i^N \|(\ddot{y} - \ddot{y}_{cmd})_i\|_{W_i}^2 \tag{5.13}$$

$$\text{s.t.} \quad M\dot{v} + C = u + J_\lambda^T \lambda \tag{5.14}$$

$$u_{min} \leq u \leq u_{max}. \tag{5.15}$$

Using this general OSC formulation, we explicitly define the tracking objectives as follows. The time-varying objectives of the OSC are the end effector position trajectory $q_{ee}(t)$ and the end effector force trajectory $u_{lcs}(t)$, so $y_{des,0}(t) = q_{ee}(t)$ and corresponding derivatives and $\lambda_{ee} = u_{lcs}(t)$. For the other objectives, the end effector orientation target is the neutral quaternion $y_{des,1} = [1, 0, 0, 0], \dot{y}_{des,1} = \ddot{y}_{des,1} = [0, 0, 0]$ because we assume the end effector can only move in translation degrees of freedom. Additionally, in order to keep the robot arm in the "elbow down" configuration and have a unique robot configuration for a given end effector position and orientation target, we specify a single joint-space tracking objective to keep the second joint of the arm at a fixed angle.

End Effector Force Target

The end effector force target is an important component to accurately tracking the MPC plan without relying on overly stiff impedance gains or an integral term, both of which could cause instability for this task. To see this, consider the scenario where the robot balances the tray. Without a force target, the robot will not compensate for the weight of the object, and the object will sag according to the impedance stiffness K_p. While tracking error for interactions solely between the manipulator and object scales with stiffness, tracking error for systems with additional contacts is more complex. For example during the sliding maneuver, even small forces applied by the end effector to the object can result in significant effects on the weight distribution of the object across the supports and end effector. Because our task is governed by friction forces, this objective is particularly important. In other words, the end effector force target is the feedforward component of the control input from the MPC, which is understandably critical when tracking dynamic motions.

5.6. Experiments

5.6.1. Task Parameters

The exact position targets are given in Table 5.1. The positions are defined in the world frame where the robot base is at the origin. For the first and second targets, the end effector target is the same as the tray, just offset in the vertical position to compensate for the thicknesses. Critically, the third target for the tray is outside the workspace limits of the end effector, so for this pair of targets the end effector target is chosen as the closest point within the feasible workspace. We choose to

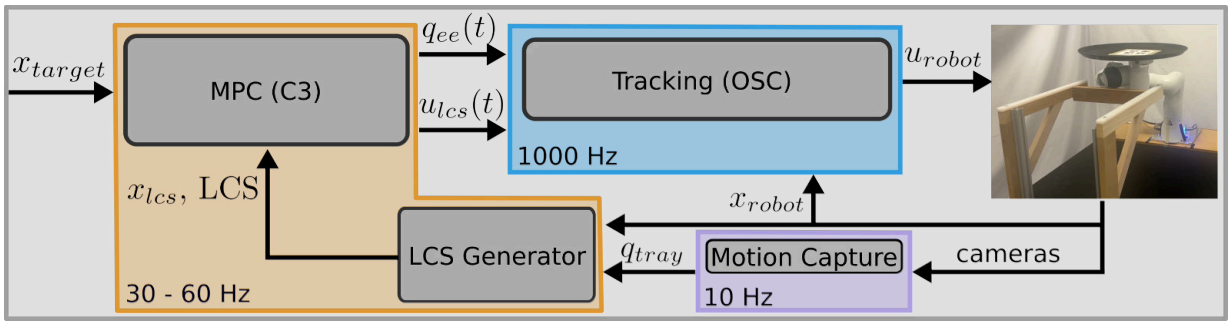


Figure 5.5: System diagram for the hardware implementation. The different colored boxes indicate separate processes which are connected via arrows that indicate represent communication via ROS/LCM.

make the third target to be identical to the initial position in order to be able to repeatedly execute the experiment without manually resetting the task environment. As detailed in Section 5.3, the next target is only given when the tray reaches the previous target. We define reaching the target as being within 5 cm from the target location.

	Tray (m)	End Effector (m)	Idle Time (s)
Initial Position	[0.7, 0.0, 0.485]	[0.55, 0, 0.45]	
First Target	[0.45, 0, 0.485]	[0.45, 0, 0.47	0.5
Second Target	[0.45, 0, 0.60]	[0.45, 0, 0.585]	3.0
Third Target	[0.7, 0, 0.485]	[0.6, 0, 0.47]	

Table 5.1: Target positions for tray retrieval task. Positions are specified as meters and in the robot/world frame where the base of the robot is at the origin [0, 0, 0]. Idle time indicates how long the robot must remain at the target before the next target is given.

Tray and End Effector

We use a standard circular food service tray with a smooth low friction bottom surface and a rubberized high friction upper surface. We model the tray as a cylinder with uniform density. We machine the disk-shaped end effector out of aluminum. Because the coefficient of friction between the machined aluminum and the tray's bottom surface is not sufficiently high, we cover the top surface of the end effector with tape. We estimate the friction coefficients by slowly tilting the supports or end effector until the tray slips and using that angle to determine a single friction coefficient, assuming that the static and dynamic coefficients are identical. Detailed parameters for both objects are listed in Table 5.2 and the objects are shown in Fig. 5.6.

	Value
Tray Mass	1 kg
Tray Radius	0.228 m
Tray Thickness	0.004 m
Tray Height (including raised rim)	0.022 m
End Effector Mass	0.37 kg
End Effector Radius	0.0725 m
End Effector Thickness	0.01 m
Tray/Support Friction Coefficient	0.18
Tray/End Effector Friction Coefficient	0.5

Table 5.2: Physical parameters for tray manipulation task.

Franka Panda

Communication between the low-level OSC controller and the Franka Panda was handled by a direct torque passthrough controller written using franka ros, a ROS wrapper around libfranka. We receive joint state messages from and send joint torques commands to the robot at 1000 Hz. A separate LCM and ROS bridge is dedicated to translating between message types. Notably, in franka ros, it was necessary to relax the torque and force thresholds from their default limits in order to accommodate the fast motions and interaction forces in this task.

5.6.2. Implementation

Both C3 and the OSC were implemented in C++ with the help of the Drake robotics library (Tedrake and the Drake Development Team, 2019). Both controllers, as well as franka ros and the LCM to ROS bridges, are run on the same desktop with a Intel i7-8700K processor. The QP step of C3 was solved using OSQP (Stellato et al., 2020), while the MIQP projection was solved using Gurobi (Gurobi Optimization, LLC, 2023). The OSC QP was solved using OSQP (Stellato et al., 2020) at 1000 Hz. We tune the OSC and C3 parameters by executing the task in the Drake (Tedrake and the Drake Development Team, 2019) simulator using the hydroelastic contact model (Masterjohn et al., 2022). We directly apply the parameters that were tuned in simulation on hardware without additional tuning.

We chose $N = 5$ knot points, a timestep of 0.075s for a time horizon of 0.3 s, and 2 ADMM iterations for each C3 solve. Under this choice of C3 parameters, we receive a new plan between

30 - 60 Hz. The friction coefficient for the contacts between the tray and end effector was set to $\mu_{tray,ee} = 0.6$ and the friction coefficient for the contacts between the tray and the supports was set to $\mu_{tray,supports} = 0.1$. The full set of C3 parameters are explained and listed in Section 5.6.3

Motion Capture

We use an off-the-shelf motion capture system (Pfrommer and Daniilidis, 2019), which uses AprilTags attached to the tray to publish the position of the tray via ROS at 10 Hz. The placement of the cameras is shown in Fig. 5.7.

Figure 5.6: End effector attached to Franka robot and serving tray with attached AprilTag.

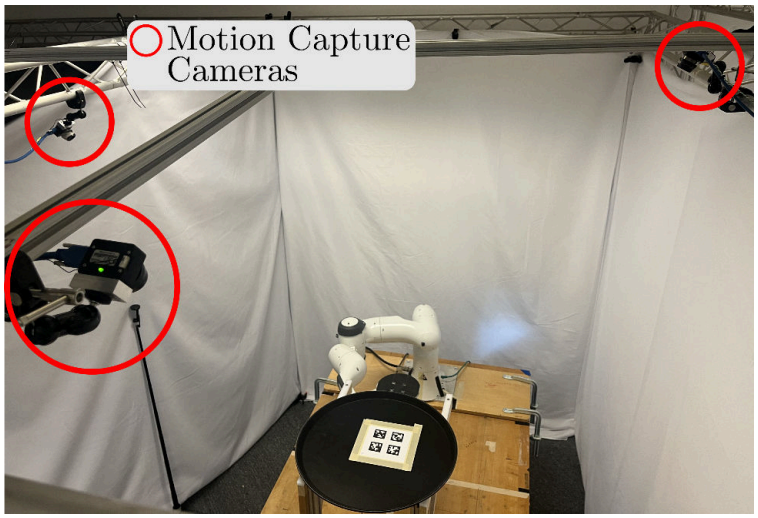


Figure 5.7: We use three cameras mounted above the task setup along with AprilTags to estimate the pose of the tray. Multiple cameras and AprilTags are used for redundancy and to reduce the effects of occlusions.

5.6.3. Description of C3 Parameters

The full set of C3 parameters for the tray retrieval task is reported in Table 5.3. N is the MPC horizon, dt is the planning timestep, $\mu_{\cdot,\cdot}$ is the single friction coefficient for geometry pairs, and ρ is an ADMM parameter. The matrices Q, R, G, U are all diagonal matrices, so we report the diagonal terms when indicated for conciseness. Note, we do not define a separate terminal cost, i.e. $Q_f = Q$. In Table 5.3, we separate Q into Q_q and Q_v, where $Q = \begin{bmatrix} \text{diag}(Q_q) & \\ & \text{diag}(Q_v) \end{bmatrix}$. G and U are the cost matrices used to establish convergence between the MPC and projection step. We use only three values to parameterize G and U, with a value corresponding to the weights for the state variables, contact variables, and input variables respectively. Thus, the G and U are constructed from three diagonal matrices as $G = \begin{bmatrix} G_x & & \\ & G_\lambda & \\ & & G_u \end{bmatrix}$, where $G_x = w_{G_x} I$. Workspace limits are imposed only on the end effector position as $lb \leq A_{workspace} q_{ee} \leq ub$. For simplicity, we set $A_{workspace} = I$ and $lb = q_{ee,min}$, $ub = q_{ee,max}$.

N	5
dt	0.075
$\mu_{tray,ee}$	0.6
$\mu_{tray,supports}$	0.1
ρ	4
ADMM iterations	2
Q_q	50 * [150, 150, 150, 0, 1, 1, 0, 15000, 15000, 15000]
Q_v	50 * [5, 5, 15, 10, 10, 1, 5, 5, 5]
R	50 * [0.15, 0.15, 0.1]
w_{G_x}	0.1
w_{G_λ}	10
w_{G_u}	0.1
w_{U_x}	0.1
w_{U_λ}	10
w_{U_u}	3
u_{min}	[-10, -10, 0]
u_{max}	[10, 10, 30]
$q_{ee,min}$	[0.4, -0.1, 0.35]
$q_{ee,max}$	[0.6, 0.1, 0.7]

Table 5.3: Full C3 parameters used across all tray retrieval experiments

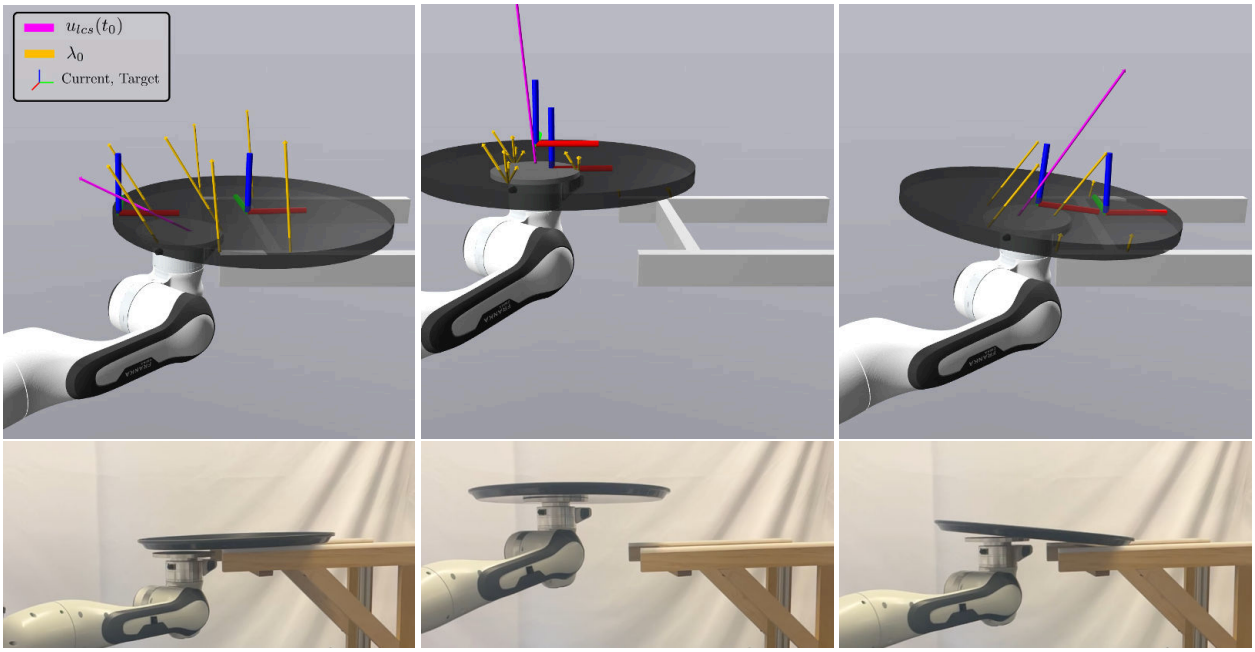


Figure 5.8: Examples of the MPC plan for retrieval (left), lift (center), and place (right), where the current state and target state of the tray are represented as triads. The MPC plans the states, inputs, and forces for N timesteps into the future. The forces (yellow arrows) and inputs (pink arrow) at the first timestep are visualized for each maneuver. For the retrieval maneuver, the plan heavily relies on the external supports compared to the other two maneuvers, where the primary contacts forces are between the end effector and tray.

5.7. Results

We performed multiple experiments to validate the robustness and generality of our framework. First we ablate our design decision to include the force tracking objective in the OSC by running experiments with and without that objective. We then demonstrate the reliability by continuously executing the experiment without manual resetting. Then, we demonstrate the robustness of our method to inaccurate models of mass and inertia by placing objects on the tray. We use the same controller parameters for all three targets and across all the demonstrations, and footage of the experiments can be seen in the accompanying website video[7].

[7] https://dynamic-controlled-sliding.github.io/

5.7.1. Force Tracking Ablation

First we ablate the contribution of tracking the end effector force by executing 10 experiments with and without the tracking objective. The tracking controller with the end effector force objective succeeded for 80% of the trials, failing once when trying to reach the second target when lifting an unbalanced tray and failing once to reach the third target when the tray slipped off in the direction of the robot base. The trajectories of the end effector and tray for an execution are shown in Fig. 5.9 and Fig. 5.11. The tracking controller without the end effector force objective succeeded for 30% of the trials, failing seven times to reach the third target due to the tray either colliding when the supports or slipping off in the direction of the robot base.

5.7.2. Reliability Test

The reliability of our method is evaluated by repeatedly executing the task without intervention. This is possible without manual resetting because the final target position coincides with the initial position, and thus we treat tracking error from the previous execution as unstructured perturbations to the initial state. Our method was able to complete six full cycles before failing due to the tray reach the position threshold (within 5cm) of the third target.

5.7.3. Task Variations

To evaluate the robustness of our method to model error, specifically inaccurate mass and inertia properties, we add two different objects on top of the tray as shown in Fig. 5.10. The first object is a common household mug that weighs 0.319 kg ($\sim$30% mass of the tray). We place it at an arbitrary position on the tray but take care to not obstruct the AprilTags on the tray. We similarly test the tray with the second object, which is a sugar box that weighs 0.515 kg ($\sim$50% mass of the tray). Without adjusting any parameters, our controller is able to successfully complete the full task without failure.

Although our method demonstrates robustness to moderate model error, it fails when we double the mass by stacking two trays. However, we can adjust our controller to accommodate the two stacked trays by updating its model to reflect its new mass, inertia, and contact geometry. Specifically, this

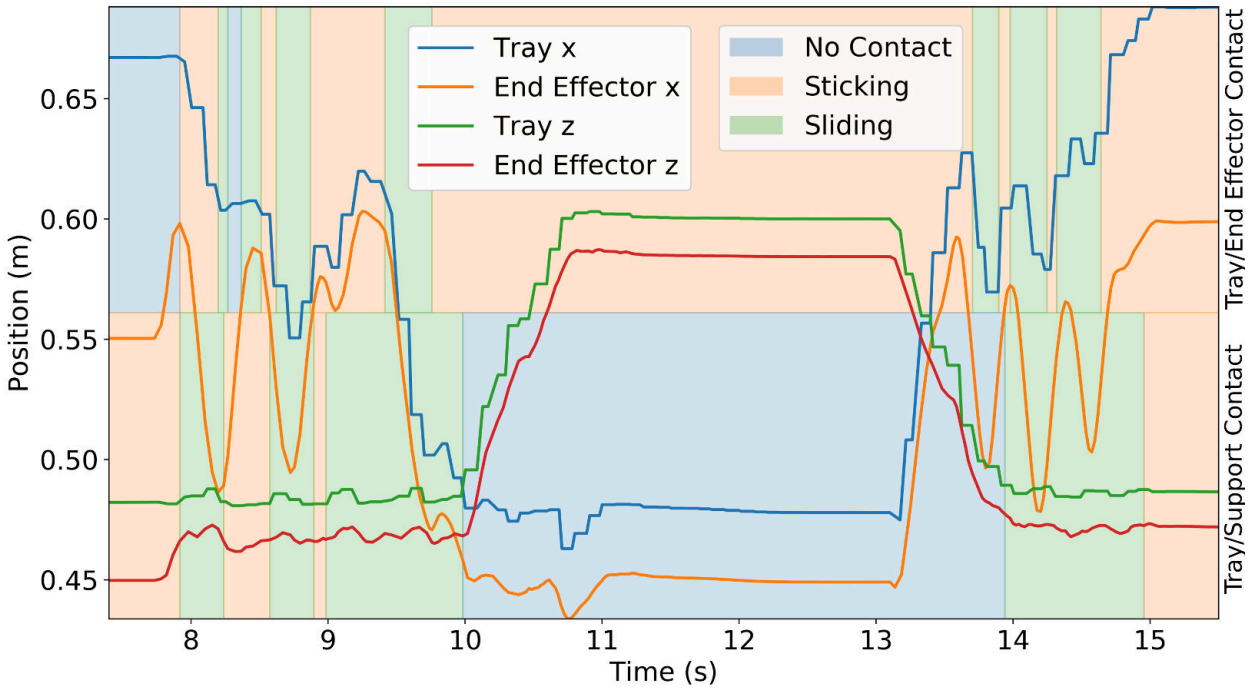
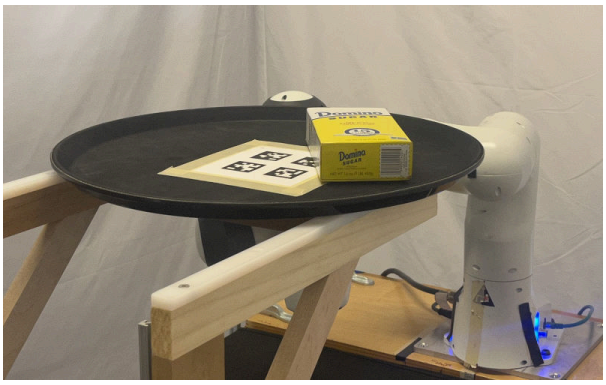


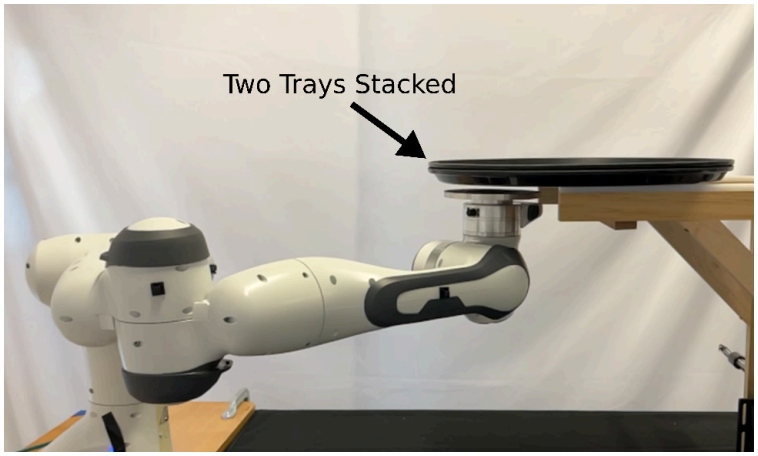
Figure 5.9: Position trajectories from execution on hardware. The visually estimated contact mode between the tray and end effector as well as tray and supports are indicated. Note, determining the actual contact mode for each of the seven contacts is challenging and there are likely many more contact mode transitions than reported in the figure. For instance, as shown in the last frame of Fig. 5.8, each contact point can be active independently.

means updating the corresponding values in the URDF file that defines the tray model. The task demonstrates the, perhaps obvious, finding that MPC is able leverage new object models.

(a) Unmodeled mug



(b) Unmodeled sugar box



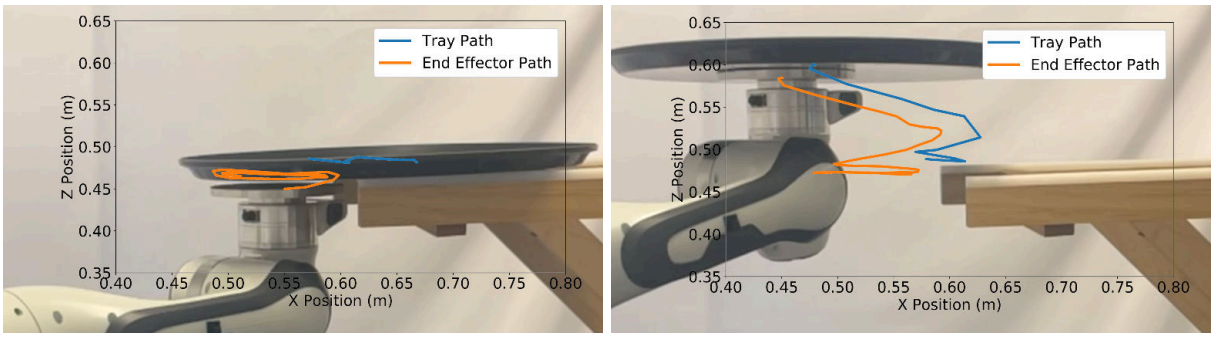
(c) Two stacked trays (modeled in the controller)

Figure 5.10: We evaluate our controller with the tray carrying unmodeled household objects placed at arbitrary positions as well as with two trays stacked on each other.

5.7.4. Behavior Analysis

We empirically observe that we did not need to tune any parameters, including friction, when transferring to hardware. We hypothesize that controller feedback and the stick-slip “gait" that naturally emerges from MPC has some inherent robustness to minor over *and* under estimation of friction. As evidence for this hypothesis, we observe the trajectory traces of the end effector and tray for two sections of the task where transitions between sliding and sticking contact are prevalent. The first section is during the retrieval task when the controller attempts to slide the tray onto the end effector. We plot a 1.5 second trajectory of the initial retrieval maneuver in Fig. 5.11, which

shows that the end effector is not only moving back and forth along the direction of the target, but also raising and lowering in a circular pattern. This gait increases the normal force between the tray when attempting to stick and decreases the normal force when attempting to slide, even utilizing the supports to entirely break contact with the tray. This difference in contact forces results in a margin for the boundary between sticking and sliding. The second section is during the place task when the controller attempts to slide the tray off of the end effector back onto the supports and is also shown in Fig. 5.11. Here, the controller accelerates the tray forward and down in order to initiate sliding followed by a similar gait pattern as the first target once the tray is on the supports. Although underestimating the friction force may not lead to failure as the tray should still reach the supports, overestimating the friction force may cause sliding during the initial forward acceleration. This may explain why, during the ablation study, the most frequent failures were during this maneuver.



(a) Retrieve Target (b) Place Target

Figure 5.11: Portions of the end effector and tray trajectories approximately overlaid on top of image showing the naturally planned gaits from the MPC. The selected trajectory for the retrieve target (a) is 1.5 seconds long and 2.5 seconds long for the place target (b).

5.7.5. Perturbation Recovery

The predominant motion of the task is along the x and z-axes. To showcase the 3D nature of our method and to highlight its reactivity properties, we apply manual perturbations directly to the tray primarily along the y-axis during execution of the experiment. Our controller is able to recover from modest perturbations applied during execution. Footage of these perturbation recoveries are included in the supplemental video.

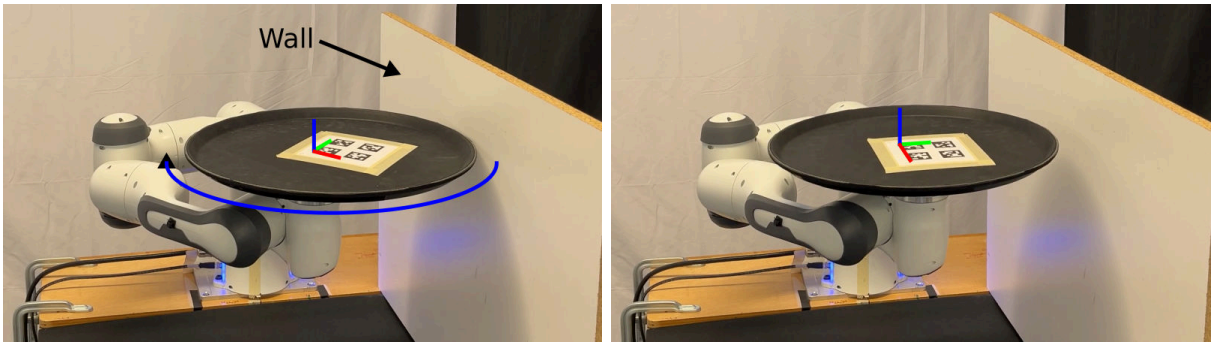


Figure 5.12: We apply our framework to a different task where the robot is tasked with rotating the tray with the aid of a wall. Using the same LCS model for the end effector and tray and a single set of gains and a single target, our framework is able to successfully accomplish the task. The system is initialized (left) so that the tray must be rotated by approximately -45 degrees about the z-axis in order to reach the target configuration (right).

5.7.6. Tray Rotation using an External Wall

To showcase the generality of our framework, we consider a different task where the robot is initialized with the tray balanced on the end effector and must rotate the tray using an external wall placed to the side of the robot as shown in Fig. 5.12. Reorienting objects has many practical uses such as changing the viewing angle of the tray. This task has similarities to previous works (Chavan-Dafle et al., 2020) (Hou et al., 2018) that use external contacts to reorient an object grasped within parallel jaw grippers. However, in our variation of the reorientation task, the object has no corners to use a pivot points. In fact, only the tangential component from the external wall contact applies a useful moment for rotating the tray. Additionally, because the tray is balanced on the end effector and not rigidly grasped, we have limited control authority to adjust the contact forces between the end effector and the tray. The result is an underactuated reorientation task that requires careful planning and control of the sliding forces on the end effector and rolling between the tray and wall.

We are able to use the same representation for the end effector and tray in the LCS. The only difference in the LCS model is the removal of the four point contacts from the supports, and the additional of a single point contact to capture the collision between the tray, which is modeled as a cylinder, and the wall, which is modeled as a box. Both of these geometries are convex geometry

primitives supported by the Drake geometry engine, and therefore no additional engineering is required.

We show again using a single set of MPC gains and a single target position for the tray and end effector, that our framework successfully moves the tray to contact the wall and rotate the tray by 45 degrees around the z-axis in either direction. The single fixed target is set to

$$q_{lcs,des} = [0.55, 0.0, 0.469, 1, 0, 0, 0, 0.55, 0.0, 0.485],$$

for this task and the state of the system is initialized to approximately

$$q_{lcs,init} = [0.55, 0.0, 0.469, 0.925, 0, 0, 0.38, 0.55, 0.02, 0.485].$$

The difference in the target and initial state is in the orientation components , which is approximately a rotation of 45 degrees about the world z-axis. For the other direction, we keep the target state values fixed, but initialize the system so that the tray is instead rotated by -45 degrees about the same world z-axis. Another difference in the target and initial state is a small offset of approximately 2 cm in the initial $tray_y$ position toward the direction of the wall. This is to encourage the MPC to utilize the wall, because otherwise it would have to trade off position error with orientation error instead of reducing both simultaneously. Additionally, the wall is placed near the side of the robot in order to decrease the penalty of using the wall, as the robot would need to move away from the target in order to make contact with the wall. Finally, because the state tracking terms of the MPC error $(x_{lcs,des} - x_{lcs})^T Q(x_{lcs,des} - x_{lcs})$ is improperly defined for the quaternion components, we compute the quaternion orientation error expressed in angle-axis form and set the desired tray angular velocity to be proportional to that error. We report the full set of MPC parameters in Table 5.4.

In the hardware setup, we use a single particle board as the wall and use the same tray as the tray retrieval task. Additionally, we add high friction tape to the rim of the tray in order to increase the friction of that surface.

N	4
dt	0.05
$\mu_{tray,ee}$	0.8
$\mu_{tray,wall}$	1.0
ρ	5
ADMM iterations	3
Q_q	50 * [10, 10, 150, 1000, 1000, 1000, 1000, 25, 25, 15000]
Q_v	50 * [5, 5, 5, 1, 1, 500, 5, 5, 5]
R	75 * [1.9, 0.5, 0.05]
w_{G_x}	0.5
w_{G_λ}	75
w_{G_u}	1.25
w_{U_x}	0.5
w_{U_λ}	50
w_{U_u}	15
u_{min}	[-10, -10, 0]
u_{max}	[10, 10, 30]
$q_{ee,min}$	[0.45, -0.2, 0.4]
$q_{ee,max}$	[0.7, 0.2, 0.5]

Table 5.4: Full C3 parameters used for rotating with external wall experiment

5.8. Limitations

While our controller is fairly robust to mass and inertia, it is not robust to the height of the supports. This is not surprising as the tray, supports, and robot are stiff, leading to sensitivity at the boundary between contact and no-contact. However, this can be addressed by quickly adapting the LCS parameters (Huang et al., 2023) or generating the contact geometry and system dynamics entirely (Howell et al., 2022b) (Pfrommer et al., 2021). We also must manually define the 3 contact points on the end effector instead of automatically generating the contacts from the end effector geometry. This was necessary to capture the surface-surface contact between the end effector and tray. An alternate approach would be to model the interactions using a single patch contact, but patch contact models have not been shown to work in a LCS. However, patch contacts can be captured by an equivalent single point contact at the center of pressure (Le Cleac'h et al., 2023) (Dietz et al., 2024). This may be a promising approach to not only reduce the size of the problem, but avoid manually defining the contact points.

Another limitation of this method originates from the LCS model used by the MPC. A known limitation of the LCS model is that it only is a linear representation of the contact decisions, meaning that it cannot consider contacts beyond the contact boundary. For example, if the end effector moves entirely out from underneath the tray, the gap function ϕ will shift and instead

consider only horizontal contact between the end effector and tray. In this scenario, the MPC cannot find a solution for the task as it cannot reason about a path to go back under the tray. This linear representation of contact boundaries can be limiting when applying this method to certain dynamic manipulation tasks such as flipping or scooping, however this can be addressed by linearizing about a reference trajectory with more informative configurations. On the other hand, dynamic manipulation tasks such as batting and throwing should, in principle, fit well within our method.

While not needing to tune a separate set of parameters for each target highlights the flexibility of our approach, our selected parameters choose to favor broad motions over fine adjustments which results in higher steady state error. An adaptive set of parameters depending on the task could reduce this tradeoff. Additionally, the success of this task is extremely sensitive to the C3 parameters. We found that many parameters need to be within 20% of their final values. Fortunately, the parameters that perform well in simulation also perform well on hardware as we did not adjust any parameters on hardware except to calibrate the height of the supports.

Finally, the tasks we consider have relatively few contacts compared to the dexterous tasks demonstrated with robotic hands (Ma et al., 2023). The number of contact variables scales linearly with the number of contacts and the possible MIQP branches per knot point scaling as 2^{n_λ}. However, in practice, high performance can often be achieved without convergence of the ADMM iterations. As a result, the compute time required to achieve sufficient performance is better than this worst-case analysis would imply. However, this also means that the compute time is sensitive to the nuances in a given task, which might lead to faster or slower convergence to a high-quality policy. We leave exploring the computational limits of MPC for future work.

5.9. Future Work

This paper demonstrates state-of-the-art performance for a on-palm sliding task, showing what can be accomplished with state feedback and contact-implicit MPC. We evaluated just one contact-implicit MPC formulation (Aydinoglu et al., 2024) on our dynamic manipulation. Many distinct formulations exist (Kurtz et al., 2023; Pang et al., 2023; Howell et al., 2022a; Le Cleac'h et al., 2024),

each with their own simplifications and assumptions. These methods have been compared on metrics such as the degrees of freedom and update rates (Kurtz et al., 2023); however, these metrics poorly capture the key factors that inform if a method will succeed. For example, the update rates are not reported on a standardized processor and are also sensitive to software optimizations independent of the algorithm. Additionally, the scaling of these methods with respect to the degrees of freedom may differ greatly. Sampling-based MPC such as Howell et al. (2022a) samples actions, which may perform poorly in large action spaces. In contrast, Aydinoglu et al. (2024) requires enumeration of all potential contact pairs, which may struggle in dexterous tasks involving robot hands. Beyond scaling, key differences may also lie in how these methods discover contacts. Comprehensive comparisons of these methods may provide instructions on how to choose between methods and inform future directions. Additionally, existing formulations seem to share common limitations, particularly the vulnerability to local minima. Future work can address how to intelligently integrate higher-level planners into this framework.

CHAPTER 6

Conclusions and Future Directions

In this thesis, we examine the challenges for control of dynamic legged locomotion and manipulation and provide methods for control *across* contact modes. We embrace the full complexity of impact dynamics and frictional contact and concede that uncertainties in measurements during impact events and the predictive performance of frictional contact are impractical to completely eliminate. Despite these concessions, we leverage understanding of the structure in these uncertainties to provide robustness to uncertainty in impact events while minimally sacrificing control authority. We apply our method to enable the first model-based jumping and running controllers for the bipedal robot, Cassie. To overcome the inaccuracies in rigid frictional contact models, we leverage the real-time ability of contact-implicit MPC to plan across contact modes without pre-specification. We demonstrate new capabilities in controlled sliding and show the promise of utilizing model-based control to reason about contact-rich manipulation without ad-hoc engineering.

A universal aspect of controlling the dynamic motions discussed this thesis is the careful consideration of the timescales under which the physical phenomena and optimization algorithms resolve. While the millisecond rates at which impact events may not be relevant to high level footstep planning decisions, it directly interacts low-level processes such as contact estimation, tracking controllers, and even motor dynamics. Similarly, we confront that the system evolves when computing the next MPC plan and how that may reduce the relevance of the plan and affect the efficacy of common computational optimization such as caching and warm-starting.

6.1. Future Work

While the results in this thesis demonstrate impressive new capabilities in the context of dynamic robots, the controllers are still not reliable enough to confidently deploy in the world. One particularly important shortcoming is the challenge of generating good models. Many of the sim-to-real challenges implementing the methods in this thesis can be classified under model curation. Beyond accurate system identification, understanding and determining when additional model fidelity was

required was an intensive effort. Additionally, the predominant workflow when engineering robotic systems is sequential: first comprehensive modeling and system identification is manually performed by engineers, which is later consumed by the controller. In order to function as true autonomous systems that can be deployed in complex environments, robots must be able to synthesize these models automatically. Thus additional work needs to be done to design controllers that can intelligently collect data while simultaneously being robust to model uncertainty due to lack of data. While adaptive controllers (Huang et al., 2023) are a promising step in this direction, an important future direction is investigating how to safely perform adaptive methods when the consequences of incorrect actions are more severe.

BIBLIOGRAPHY

Brian Acosta, William Yang, and Michael Posa. Validating robotics simulators on real-world impacts. *IEEE Robotics and Automation Letters*, 7(3):6471–6478, 2022.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

Mihai Anitescu and Florian A Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, 1997. URL https://link.springer.com/article/10.1023/A:1008292328909.

C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padir, F. Polido, G. G. Tighe, and X. Xinjilefu. No falls, no resets: Reliable humanoid behavior in the darpa robotics challenge. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 623–630, 2015. doi: 10.1109/HUMANOIDS.2015.7363436.

Alp Aydinoglu, Adam Wei, Wei-Cheng Huang, and Michael Posa. Consensus complementarity control for multi-contact mpc. *IEEE Transactions on Robotics*, 2024.

Maria Bauza, Francois R Hogan, and Alberto Rodriguez. A data-efficient approach to precise and controlled pushing. In *Conference on Robot Learning*, pages 336–345. PMLR, 2018.

Gerardo Bledt. *Regularized predictive control framework for robust dynamic legged locomotion*. PhD thesis, Massachusetts Institute of Technology, 2020. URL https://dspace.mit.edu/handle/1721.1/125485.

Gerardo Bledt and Sangbae Kim. Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6316–6323. IEEE, 2019.

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, 2018a.

Gerardo Bledt, Patrick M Wensing, Sam Ingersoll, and Sangbae Kim. Contact model fusion for event-based locomotion in unstructured terrains. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018b.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Zachary Brei, Jonathan Michaux, Bohao Zhang, Patrick Holmes, and Ram Vasudevan. Serving time: Real-time, safe motion planning and control for manipulation of unsecured objects. *IEEE Robotics and Automation Letters*, 9(3):2383–2390, 2024. doi: 10.1109/LRA.2024.3355731. URL https://ieeexplore.ieee.org/document/10403905.

Anindya Chatterjee. *Rigid body collisions: some general considerations, new collision laws, and some experimental data*. Cornell University, 1997.

Nikhil Chavan-Dafle, Rachel Holladay, and Alberto Rodriguez. Planar in-hand manipulation via motion cones. *The International Journal of Robotics Research*, 39(2-3):163–182, 2020. URL https://journals.sagepub.com/doi/full/10.1177/0278364919880257.

Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84), 2023. URL https://www.science.org/doi/abs/10.1126/scirobotics.adc9244.

Yu-Ming Chen and Michael Posa. Optimal reduced-order modeling of bipedal locomotion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8753–8760. IEEE, 2020.

Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T. Mason. Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2730–2736. IEEE, 2022. URL https://ieeexplore.ieee.org/document/9811872.

Xianyi Cheng, Sarvesh Patil, Zeynep Temel, Oliver Kroemer, and Matthew T Mason. Enhancing dexterity in robotic manipulation via hierarchical contact exploration. *IEEE Robotics and Automation Letters*, 9(1):390–397, 2023. URL https://ieeexplore.ieee.org/document/9811872.

Christine Chevallereau, Gabriel Abba, Yannick Aoustin, Franck Plestan, Eric Westervelt, Carlos Canudas De Wit, and Jessy Grizzle. Rabbit: A testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *2023 Robotics Sciences and Systems*, 2023. URL https://roboticsconference.org/2023/program/papers/026/.

Matthew Chignoli, Savva Morozov, and Sangbae Kim. Rapid and reliable quadruped motion planning with omnidirectional jumping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6621–6627. IEEE, 2022.

Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336345. doi: 10.1145/2776880.2792704. URL https://doi.org/10.1145/2776880.2792704.

Aidan Curtis, Xiaolin Fang, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Caelan Reed Garrett. Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1940–1946. IEEE, 2022. URL https://ieeexplore.ieee.org/document/9812057.

Hongkai Dai and Russ Tedrake. Optimizing robust limit cycles for legged locomotion on unknown terrain. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1207–1213. IEEE, 2012.

Monica A Daley and Andrew A Biewener. Running over rough terrain reveals limb control for intrinsic stability. *Proceedings of the National Academy of Sciences*, 103(42):15681–15686, 2006.

Christian Dietz, Armin Nurkanović, Sebastian Albrecht, and Moritz Diehl. High accuracy numerical optimal control for rigid bodies with patch contacts through equivalent contact points–extended version. *arXiv preprint arXiv:2403.13931*, 2024.

Neel Doshi, Orion Taylor, and Alberto Rodriguez. Manipulation of unknown objects via contact configuration regulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2693–2699, 2022. doi: 10.1109/ICRA46639.2022.9811713. URL https://ieeexplore.ieee.org/document/9811713.

Oluwami Dosunmu-Ogunbi, Aayushi Shrivastava, Grant Gibson, and Jessy W Grizzle. Stair climbing using the angular momentum linear inverted pendulum model and model predictive control. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8558–8565. IEEE, 2023.

Luke Drnach and Ye Zhao. Robust trajectory optimization over uncertain terrain with stochastic complementarity. *IEEE Robotics and Automation Letters*, 6(2):1168–1175, 2021.

Boston Dynamics. Blog: Flipping the script with atlas, 2021. URL https://bostondynamics.com/blog/flipping-the-script-with-atlas/.

Nima Fazeli, Samuel Zapolsky, Evan Drumwright, and Alberto Rodriguez. Fundamental limitations in performance and interpretability of common planar rigid-body contact models. In *International Symposium on Robotics Research (ISRR)*, pages 555–571. Springer, 2020.

Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014. URL https://link.springer.com/book/10.1007/978-1-4899-7560-7.

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.

Grant Gibson, Oluwami Dosunmu-Ogunbi, Yukai Gong, and Jessy Grizzle. Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6724–6731.

IEEE, 2022.

Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988. URL https://ieeexplore.ieee.org/document/2083.

Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

Yukai Gong and Jessy W Grizzle. Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion. *Journal of Dynamic Systems, Measurement, and Control*, 144(12):121006, 2022.

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.

Bernhard P Graesdal, Shao YC Chia, Tobia Marcucci, Savva Morozov, Alexandre Amice, Pablo A Parrilo, and Russ Tedrake. Towards tight convex relaxations for contact-rich manipulation. *arXiv preprint arXiv:2402.10312*, 2024.

Kevin Green, Ross L Hatton, and Jonathan Hurst. Planning for the unexpected: Explicitly optimizing motions for ground uncertainty in running. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1445–1451. IEEE, 2020.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

Mathew Halm and Michael Posa. Modeling and analysis of non-unique behaviors in multiple frictional impacts. In *Robotics: Science and Systems*, 2019.

Ross Hartley, Maani Ghaffari, Ryan M Eustice, and Jessy W Grizzle. Contact-aided invariant extended kalman filtering for robot state estimation. *The International Journal of Robotics Research*, 39(4):402–430, 2020.

WPMH Heemels, Johannes M Schumacher, and S Weiland. Linear complementarity systems. *SIAM journal on applied mathematics*, 60(4):1234–1269, 2000. URL https://doi.org/10.1137/S0036139997325199.

Adam Heins and Angela P. Schoellig. Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator. *IEEE Robotics and Automation Letters*, 8(12):7986–7993, 2023. doi: 10.1109/LRA.2023.3324520. URL https://ieeexplore.ieee.org/document/10285028.

Mitsuru Higashimori, Keisuke Utsumi, Yasutaka Omoto, and Makoto Kaneko. Dynamic manipu-

lation inspired by the handling of a pizza peel. *IEEE Transactions on Robotics*, 25(4):829–838, 2009. doi: 10.1109/TRO.2009.2017085. URL https://ieeexplore.ieee.org/document/4814586.

Masato Hirose and Kenichi Ogawa. Honda humanoid robots development. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):11–19, 2007.

Jessica Hodgins and Marc H Raibert. Biped gymnastics. *Dynamically Stable Legged Locomotion*, 79, 1988.

N Hogan. Impedance control-an approach to manipulation. i-theory. ii-implementation. iii-applications. *ASME Journal of Dynamic Systems and Measurement Control B*, 107:1–24, 1985.

Philip Holmes, Robert J Full, Dan Koditschek, and John Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM review*, 48(2):207–304, 2006.

Yifan Hou, Zhenzhong Jia, and Matthew T Mason. Fast planning for 3d any-pose-reorienting using pivoting. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1631–1638. IEEE, 2018. URL https://ieeexplore.ieee.org/document/8462834.

Yifan Hou, Zhenzhong Jia, Aaron M Johnson, and Matthew T Mason. Robust planar dynamic pivoting by regulating inertial and grip forces. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pages 464–479. Springer, 2020. URL https://link.springer.com/chapter/10.1007/978-3-030-43089-4_30.

Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive sampling: Real-time behaviour synthesis with mujoco. *arXiv preprint arXiv:2212.00541*, 2022a.

Taylor A Howell, Simon Le Cleac'h, Jan Brüdigam, J Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable physics engine for robotics. *arXiv preprint arXiv:2203.00806*, 2022b. URL https://arxiv.org/abs/2203.00806.

Albert S Huang, Edwin Olson, and David C Moore. Lcm: Lightweight communications and marshalling. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4057–4062. IEEE, 2010. URL https://ieeexplore.ieee.org/document/5649358.

Wei-Cheng Huang, Alp Aydinoglu, Wanxin Jin, and Michael Posa. Adaptive contact-implicit model predictive control with online residual learning. *arXiv preprint arXiv:2310.09893*, 2023. URL https://arxiv.org/abs/2310.09893.

Se Hwan Jeon, Sangbae Kim, and Donghyun Kim. Online optimal landing control of the mit mini cheetah. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 178–184. IEEE, 2022.

Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 239–246. IEEE, 2001.

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

Daniel E Koditschek and Martin Buehler. Analysis of a simplified hopping robot. *The International Journal of Robotics Research*, 10(6):587–605, 1991.

Vince Kurtz, Alejandro Castro, Aykut Özgün Önol, and Hai Lin. Inverse dynamics trajectory optimization for contact-implicit model predictive control. *arXiv preprint arXiv:2309.01813*, 2023. URL https://arxiv.org/abs/2309.01813.

Simon Le Cleac'h, Mac Schwager, Zachary Manchester, Vikas Sindhwani, Pete Florence, and Sumeet Singh. Single-level differentiable contact simulation. *IEEE Robotics and Automation Letters*, 8 (7):4012–4019, 2023.

Simon Le Cleac'h, Taylor A. Howell, Shuo Yang, Chi-Yen Lee, John Zhang, Arun Bishop, Mac Schwager, and Zachary Manchester. Fast contact-implicit model predictive control. *IEEE Transactions on Robotics*, pages 1–14, 2024. doi: 10.1109/TRO.2024.3351554. URL https://ieeexplore.ieee.org/document/10384795.

He Li and Patrick M Wensing. Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, 5(4):5448–5455, 2020.

Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *arXiv preprint arXiv:2401.16889*, 2024.

Kevin M Lynch and Matthew T Mason. Dynamic nonprehensile manipulation: Controllability, planning, and experiments. *The International Journal of Robotics Research*, 18(1):64–92, 1999. URL https://journals.sagepub.com/doi/abs/10.1177/027836499901800105.

Wen-Loong Ma, Shishir Kolathaya, Eric R Ambrose, Christian M Hubicki, and Aaron D Ames. Bipedal robotic running with durus-2d: Bridging the gap between theory and experiment. In *Proceedings of the 20th international conference on hybrid systems: computation and control*, pages 265–274, 2017.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via

coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Matthew T Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

Matthew T Mason and Kevin M Lynch. Dynamic manipulation. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, volume 1, pages 152–159. IEEE, 1993.

Sean Mason, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Balancing and walking using full dynamics lqr control with contact constraints. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 63–68. IEEE, 2016.

Joseph Masterjohn, Damrong Guoy, John Shepherd, and Alejandro Castro. Velocity level approximation of pressure field contact patches. *IEEE Robotics and Automation Letters*, 7(4): 11593–11600, 2022. doi: 10.1109/LRA.2022.3203845. URL https://ieeexplore.ieee.org/document/9874987.

Tao Pang, H. J. Terry Suh, Lujie Yang, and Russ Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on Robotics*, 39(6):4691–4711, 2023. doi: 10.1109/TRO.2023.3300230. URL https://ieeexplore.ieee.org/document/10225433.

Bernd Pfrommer and Kostas Daniilidis. Tagslam: Robust slam with fiducial markers. *arXiv preprint arXiv:1910.00679*, 2019. URL https://github.com/berndpfrommer/tagslam_root.

Samuel Pfrommer, Mathew Halm, and Michael Posa. Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 2279–2291. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/pfrommer21a.html.

Quang-Cuong Pham, Stéphane Caron, Puttichai Lertkultanon, and Yoshihiko Nakamura. Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots. *The International Journal of Robotics Research*, 36(1):44–67, 2017. doi: 10.1177/0278364916675419. URL https://doi.org/10.1177/0278364916675419.

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.

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.

Ioannis Poulakakis and Jessy W Grizzle. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793, 2009.

Marc H Raibert, H Benjamin Brown Jr, and Michael Chepponis. Experiments in balance with a 3d one-legged hopping machine. *The International Journal of Robotics Research*, 3(2):75–92, 1984.

Jenna Reher and Aaron D Ames. Inverse dynamics control of compliant hybrid zero dynamic walking. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2040–2047. IEEE, 2021.

C David Remy. Ambiguous collision outcomes and sliding with infinite friction in models of legged systems. *The International Journal of Robotics Research*, 36(12):1252–1267, 2017. URL https://doi.org/10.1177/0278364917731820.

Mark Rijnen, Eric de Mooij, Silvio Traversaro, Francesco Nori, Nathan van de Wouw, Alessandro Saccon, and Henk Nijmeijer. Control of humanoid robot motions with impacts: Numerical experiments with reference spreading control. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4102–4107. IEEE, 2017.

Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. Nonprehensile dynamic manipulation: A survey. *IEEE Robotics and Automation Letters*, 3(3):1711–1718, 2018. doi: 10.1109/LRA.2018.2801939. URL https://ieeexplore.ieee.org/document/8280543.

Alessandro Saccon, Nathan van de Wouw, and Henk Nijmeijer. Sensitivity analysis of hybrid systems with state jumps with application to trajectory tracking. In *53rd IEEE Conference on Decision and Control*, pages 3065–3070. IEEE, 2014.

D. Salmon. Minimax controller design. *IEEE Transactions on Automatic Control*, 13(4):369–376, 1968. doi: 10.1109/TAC.1968.1098941.

Luis Sentis and Oussama Khatib. Control of free-floating humanoid robots through task prioritization. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1718–1723. IEEE, 2005.

André Seyfarth, Hartmut Geyer, and Hugh Herr. Swing-leg retraction: a simple control model for stable running. *Journal of Experimental Biology*, 206(15):2547–2555, 2003.

Jian Shi, J Zachary Woodruff, Paul B Umbanhowar, and Kevin M Lynch. Dynamic in-hand sliding manipulation. *IEEE Transactions on Robotics*, 33(4):778–795, 2017. URL https://ieeexplore.ieee.org/document/7913727.

Yuki Shirai, Devesh K Jha, and Arvind U Raghunathan. Robust pivoting manipulation using contact implicit bilevel optimization. *IEEE Transactions on Robotics*, 2024.

Jonah Siekmann, Yesh Godse, Alan Fern, and Jonathan Hurst. Sim-to-real learning of all common bipedal gaits via periodic reward composition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7309–7315. IEEE, 2021a.

Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *arXiv preprint arXiv:2105.08328*, 2021b.

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

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. URL https://link.springer.com/article/10.1007/s12532-020-00179-2.

David Stewart and Jeffrey C Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 162–169. IEEE, 2000. URL https://ieeexplore.ieee.org/document/844054.

Gilbert Strang. *Introduction to linear algebra*, volume 3. SIAM, 2022.

Rajesh Subburaman, Mario Selvaggio, and Fabio Ruggiero. A non-prehensile object transportation framework with adaptive tilting based on quadratic programming. *IEEE Robotics and Automation Letters*, 2023. URL https://ieeexplore.ieee.org/document/10105969.

Orion Taylor, Neel Doshi, and Alberto Rodriguez. Object manipulation through contact configuration regulation: Multiple and intermittent contacts. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8735–8743, 2023. doi: 10.1109/IROS55552.2023.10341362. URL https://ieeexplore.ieee.org/document/10341362.

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

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Jari Van Steen, Gijs Van Den Brandt, Nathan van de Wouw, Jens Kober, and Alessandro Saccon. Quadratic programming-based reference spreading control for dual-arm robotic manipulation with planned simultaneous impacts. *IEEE Transactions on Robotics*, 2024.

Jari J van Steen, Nathan van de Wouw, and Alessandro Saccon. Robot control for simultaneous impact tasks via quadratic programming-based reference spreading. In *2022 American Control Conference (ACC)*, pages 3865–3872. IEEE, 2022.

Jari J van Steen, Abdullah Coşgun, Nathan van de Wouw, and Alessandro Saccon. Dual arm impact-aware grasping through time-invariant reference spreading control. *IFAC-PapersOnLine*, 56(2):1009–1016, 2023.

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

Yuquan Wang, Niels Dehio, and Abderrahmane Kheddar. Predicting impact-induced joint velocity jumps on kinematic-controlled manipulator. *IEEE Robotics and Automation Letters*, 7(3):6226–6233, 2022.

Yuquan Wang, Niels Dehio, Arnaud Tanguy, and Abderrahmane Kheddar. Impact-aware task-space quadratic-programming control. *The International Journal of Robotics Research*, 42(14): 1265–1282, 2023.

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, 2013. doi: 10.1109/ICRA.2013.6631008. URL https://ieeexplore.ieee.org/document/6631008.

E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, 2003. doi: 10.1109/TAC.2002.806653.

J. Zachary Woodruff and Kevin M. Lynch. Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4066–4073, 2017. doi: 10.1109/ICRA.2017.7989467. URL https://ieeexplore.ieee.org/document/7989467/.

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.

James Zhu, Nathan J Kong, George Council, and Aaron M Johnson. Hybrid event shaping to stabilize periodic hybrid orbits. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 01–07. IEEE, 2022.