FLORIDA STATE UNIVERSITY

FAMU-FSU COLLEGE OF ENGINEERING

REAL-TIME PLANNING AND CONTROL OF DYNAMIC ROBOTS IN DYNAMIC ENVIRONMENTS

By

JASON WHITE

A Dissertation submitted to the
Department of Mechanical Engineering
in partial fulfillment of the
requirements for the degree of
Doctorate of Science

2023

Jason White defended this dissertation on November 10, 2023.
The members of the supervisory committee were:

Christian Hubicki
Professor Directing Thesis

Jonathan Clark
Committee Member

Brandon Krick
Committee Member

Rodney Roberts
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

ACKNOWLEDGMENTS

To my faculty mentors. Christian: Thank you for supporting me throughout my degree. Your insight and instruction have helped foster my passion for the field of robotics. Dr. Clark: Thank you for allowing me to collaborate on numerous projects, and providing sound advice when I came to you for help.

To my student colleagues. Thank you for sharing your expertise and taking the time to guide me when I first arrived with limited knowledge in the field of robotics. Thank you for spending countless hours working on projects and staying late to meet deadlines, much of my work here would not exist without your help.

To my family. None of this would be possible without you, thank you for supporting me in my career change and being there for me when I needed you.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Dynamic robots require robust controllers to successfully plan and navigate the unstructured world around us. Novel control methods become necessary when these robots' environments are dynamic and the robots themselves have fast and unstable dynamics that cannot be shaped arbitrarily by control – *e.g.*, hybrid dynamics (impacts) or underactuation. Legged robots and unmanned aerial vehicles (UAVs) are examples of such dynamic machines. This thesis argues that fast and reactive planning in dynamic environments is paramount for success when facing these challenges where control decisions must be made in milliseconds. To facilitate this goal, this work formulates new implementations of trajectory optimizations and force control as the primary tools for synthesizing legged and aerial locomotion that accommodates rapid environmental changes. As a demonstration of novel forms of dynamic planning and control, this thesis presents: 1) A bipedal robot balancing controller that remains stable in the face of unplanned contact changes (*i.e.*. soft, moving, or completely lost footholds). This feedback controller is advantageous when the dynamic model may be inaccurate, unknown, or changing, and is amenable to other robotic platforms with joint-level torque control, quickly moving limbs, and mechanical robustness to impacts. 2) A monopod motion controller that generates reactive motion plans in milliseconds by solving convex quadratic programs but does not use predefined footfall locations or contact timings. This method allows the system to hop, stand, or transition between the two as an emergent property when the robot's desired height and speed are changed. 3) A monopod motion planner that generates energy-efficient gaits in resistive fluid-like media. After modeling fluid forces on partially submerged legs as differentiable functions, gradient-based nonlinear optimizations were able to reduce computation times observed in prior work from minutes to seconds. 4) A real-time obstacle avoidance motion planner capable of avoiding multiple fast-moving obstacles both on UAV hardware and a simple bipedal locomotion model. The mathematical formulation requires few and easy-to-tune hyperparameters, allowing for straightforward implementation. Altogether, these contributions underscore how simple iteratively updated models, when coupled with fast-solving planning algorithms, yield robust and dynamic robot control.

CHAPTER 1

INTRODUCTION

1.1 Modern Robot Performance

Robots have long held the promise to mimic or exceed the reflexes, agility, and abilities of animals. Although some robots perform repetitive tasks more accurately and quickly than their human counterparts (*e.g.*, manufacturing pick-and-place tasks), unexpected circumstances can quickly lead to failed tasks or damaged hardware. This limitation has constrained the ability of robots to perform the broader scope of tasks on which our society relies. Of the available domestic robots, robotic vacuum cleaners are the most common [14]. These robots can map out and navigate around their surrounding environments but still struggle with full autonomy, frequently becoming immobilized or damaged. Commercial drones can follow an individual by using image recognition while mapping and avoiding stationary obstacles. Boston Dynamics' Atlas humanoid demonstrates impressive athleticism from dancing to back-flips and basic manipulation of large objects. However, the limitations of these commercial platforms in dynamic environments are not well-documented. For instance, would commercial drones be able to avoid adversarial or fast-moving obstacles? Similarly, how would Atlas respond to terrain that is not rigid or collapses beneath its feet? This thesis seeks to address gaps in dynamic robot control when confronted with dynamic environments such as unstable terrain, moving obstacles, legged locomotion through viscous media, and fast-legged reflexes that quickly choose new footholds (*i.e.*, through-contact planning).

1.2 Dynamic Legged Robots

Early forms of legged robots used more static control strategies to remain stable and avoid falls. For bipedal robots, a classic control approach called zero moment point [45] keeps the center of pressure safely within its polygon of foot support to prevent tipping. This conservative approach limits the range of dynamic behaviors available to the robot (*e.g.*, running). Quadrupeds use similar methods for "creeping" gaits [60] which remain stable throughout the gait by always maintaining three points of contact and a triangular polygon of support.

More modern approaches treat walking and running as dynamic tasks. Controllers are designed to withstand larger impulse/impact forces, generally synonymous with periodic stability. These gaits tend to be faster, more agile, and more energy efficient but are only intermittently stable throughout the gait cycle. Controlling these more dynamic gaits becomes challenging since impact forces are discontinuous, and footholds may not be rigid. Some of the earliest instances of dynamic locomotion can be observed in the Raibert developed in the 1980s [72]. These monopod, bipedal, and quadrupedal systems were capable of stable gait cycles and executing front flips despite having uncontrolled ballistic trajectories while airborne. More recent research showcases modern bipedal and quadrupedal robots, which are significantly larger and more complex, executing similar acrobatic maneuvers with greater precision and control [50]. Furthermore, platforms such as Anymal use the onboard perception to adapt their gait to the surroundings by tackling stairs, rough terrain, flat ground, etc., and even balancing on two legs [75].

Dynamic legged control is generally achieved using model-based controllers which leverage knowledge of the robot's system dynamics. Operational space control [6], force control [24] [1], and hybrid zero dynamics [91] are all common methods for computing the instantaneous motor torques that achieve an instantaneous body acceleration. However, these instantaneous methods cannot plan for future states, which could lead to future task failures (*e.g.*, falls). For task-based planning, trajectory optimizations are a common choice and form the foundation for planning-based controllers – notably Model-Predictive Control (MPC). Since 2019, model-free approaches such as Reinforcement Learning (RL) have successfully controlled both bipedal and quadrupedal-legged robots to achieve extreme maneuverability [58] [99]. These new deep-learning approaches create controllers that can train a policy in minutes by running thousands of simulations that tackle a broad spectrum of terrains and obstacles in parallel.

1.3 Dynamic Aerial Robots

Since the early 2000s UAVs have vastly grown in popularity with quad-copters further rising to prominence in the 2010s. These underactuated systems execute agile maneuvers with a high degree of precision while regularly subjected to external disturbances such as wind gusts. Early controllers used a backstepping control approach, a recursive algorithm that decomposes a system into subsystems that can be stabilized [57], [55]. Today, everything from linear quadratic regulators (LQR) [103] to the more recent differential flatness [61] techniques are leveraged to optimize the

control of UAVs. Other controllers such as gain scheduling [77] which uses variable gains with linear dynamics, and adaptive control [64] which automatically adjusts control parameters in real-time, embrace the power of reference tables for control. These are only a few of the options available to control engineers; neural networks [73], fuzzy logic [26], and many more can be put onto a UAV depending on its use. Many of these controllers are combined with higher-level planning which includes some form of obstacle avoidance. In most instances, these are stationary obstacles and thus encoded as a hard constraint in the planning. This thesis leverages combinations of some of these existing methods with newly presented methods to create a planner that avoids dynamic obstacles.

1.4 Challenges of Dynamic Control

Real environments are dynamic and can rapidly change. At any moment a controller may need to adjust to avoid an obstacle, react immediately to unexpected terrain, or adapt its terrain model to match a new environment. Strictly relying on a model is not always enough to overcome changes that arise in the environment. Additionally, modeling everything present in the world is not a realistic approach to control. For example, to fully model a terrestrial environment we would need all slopes, terrain, ground stiffness, ground-dampening factors, media resistance properties, etc. Developing this could take years, so instead, we would rather design a controller that adapts to its surroundings. Similarly, we would like controllers to navigate around obstacles regardless of an obstacle position, size, or speed. For these reasons, controllers that react to rapid changes rather than assume an unchanging environment are more robust in real-world unstructured settings.

Our approach minimizes the complexity of models used for planning or so-called reduced-order models. Avoiding the use of detailed models allows us to devise control computations that solve quickly, which we achieve by making non-differentiable models differentiable, linearizing system dynamics, and linearizing task definitions. Reduced-order models of obstacle avoidance, environmental media, and the robot itself are all used throughout this thesis. Reducing model complexity makes the problem faster and easier to solve for computers using linear algebra and local optimizations. After employing these techniques, the time to develop desired motor torques is reduced to < 0.5ms using force control, planning gaits through viscous media takes 1 second rather than 20 minutes, and planning trajectories around moving obstacles is accelerated to a rate of 500 solved plans per second.

1.5 Outline

This thesis is divided into four parts. Part I considers a method of reaction to dynamic terrain for a high degree of freedom biped, and introduces hardware results on the "Cassie" platform validating the proposed method. Part II develops a single-leg hopper model and method for motion planning as a convex problem without a predefined contact timing. Part III again considers a variation of dynamic terrain – planning gaits of a single-legged hopper through a viscous media, and introduces preliminary hardware results on a "Minitaur" Leg. Part IV discusses a method of avoiding dynamic adversarial obstacles, with a wide range of hardware results on a "Crazyflie" drone validating the formulation.

1.6 Contribution

The scope of this thesis includes several research topics in robot control, from optimal control to force control and legged locomotion to UAVs – most of which are validated with robot experiments on physical hardware. Much of the bipedal force controller's success stems from a lack of reliance on a complex dynamic model. By using a point-mass model and kinematic Jacobians, we were able to produce a stable force controller that is invariant to contact changes. The second topic, a through-contact legged controller, allows a monopod to hop, stand, or transition between the two by only changing the desired speed and body height. The model-predictive controller works by iteratively updating an estimated contact sequence, which when applied to monopods, emergently yields hopping due to the inevitability of footfall contact under gravity. Next, we drastically reduce the computation time for generating legged motion plans through fluid-like media by modeling the integral-based fluid forces as a continuously differentiable function. Finally, using three key components, our obstacle avoidance algorithm can avoid fast-moving dynamic obstacles on a micro UAV with extensions to bipedal walking models. To achieve agile avoidance, we penalize close encounters using a slack variable, use recent motion plans to iteratively linearize the dynamics, and estimate the obstacle's motion via first-order integration. These contributions show how models that approximate the robot and environmental dynamics, either via reduced-order formulations or by iterative estimates, allow fast computations and optimizations to generate and execute motion plans on-the-fly for dynamic systems in dynamic environments.

CHAPTER 2

BIPEDAL CONTROL ON DYNAMIC TERRAIN

Joint position control is very common in the field of bipedal robotics. However, when the terrain is soft, slippery, or moves, this method of control tends to fail. Operating in the real world requires the capacity to quickly react to these unexpected dynamic terrains. The work in this chapter presents a standing controller for the bipedal platform Cassie which compensates for soft terrain and lost or moving footholds at a rate $>$ 2000Hz. This section is a reprint of the conference paper named "Force-based Control of Bipedal Balancing on Dynamic Terrain with the Tallahassee Cassie Robotic Platform" [92] published in the 2020 International Conference on Robotics and Automation.

Specific contributions of this chapter include:

1. A bipedal controller that utilizes a minimalist dynamic model (*i.e.*, a point mass).
2. A controller that maintains balance on top of soft and rapidly-moving surfaces (*i.e.*, dynamic terrain) including foam surfaces and sudden drops.

2.1 Abstract

Out in the field, bipedal robots need to travel on terrain that is uneven, non-rigid, and sometimes moving beneath their feet. We present a force-based double support balancing controller for such dynamic terrain scenarios for bipedal robots, and test it on the robotic bipedal platform "Tallahassee Cassie." The presented controller relies on minimal information about the robot model, requiring its kinematics and overall weight, but not the inertia of individual links or components. The controller is pelvis-centric, commanding pelvis positions in Cartesian space, which a model-free PD controller converts to motor torques in joint space. By commanding forces, torques, and a frontal center of pressure in this fashion, Tallahassee Cassie is capable of balancing on a variety of scenarios, from a lifting/sliding platform, to soft foam, to a sudden drop. These results show the potential for bipedal control to balance successfully despite minimal model information, and the presence of large dynamic impacts–*e.g.*, falling through trap door, and soft series-spring deflections. These results motivate future work for walking and running controllers on dynamic terrain with relatively low reliance on modeling information.

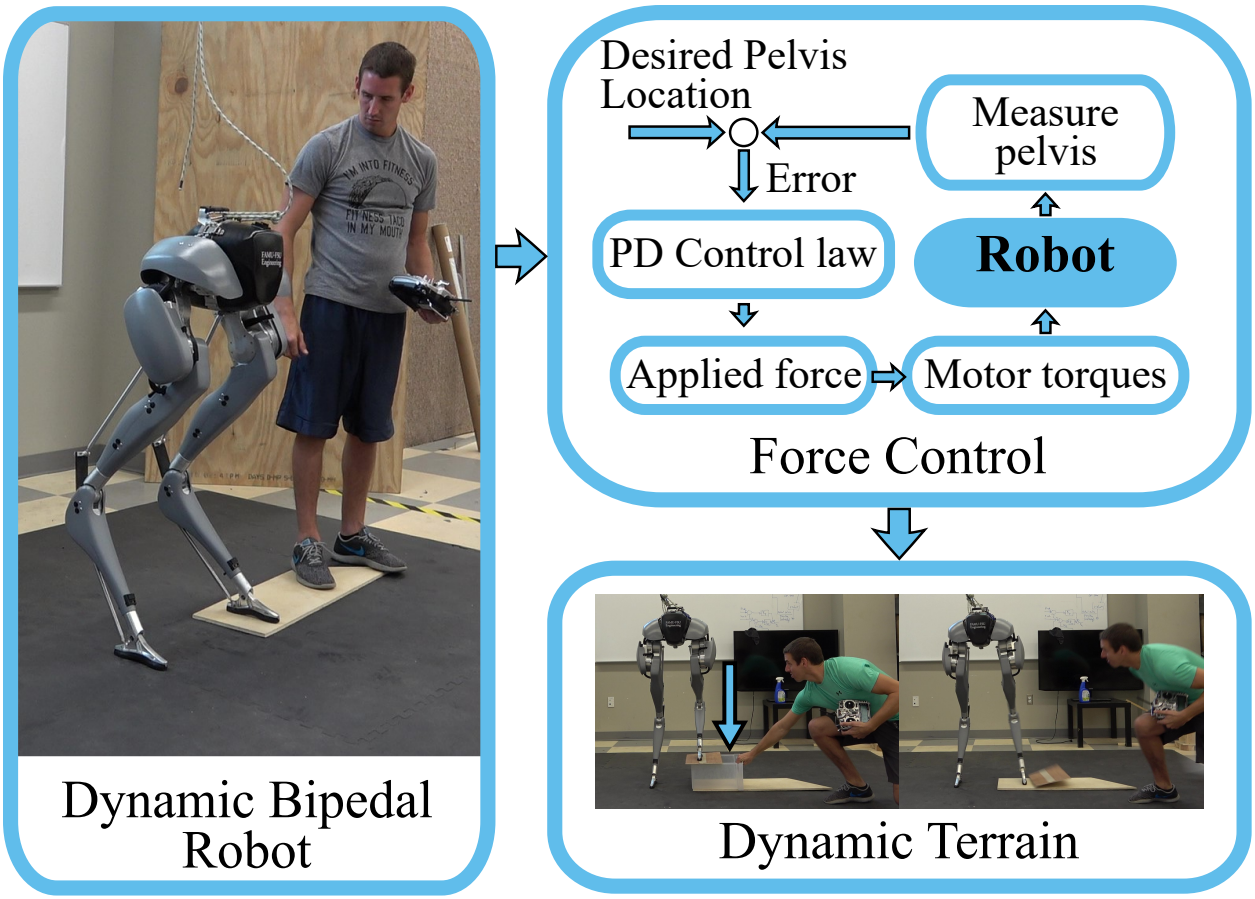

Figure 2.1: We implement a force-based controller for balancing the bipedal robot, "Tallahassee Cassie" on dynamic terrain, such as soft terrain or a sudden loss of foot contact.

2.2 Introduction

People are capable of balancing, walking, and running out in a world where terrain is often soft and yielding–*e.g.*, soil, sand, and snow. Sometimes, the ground moves by giving way beneath our feet, where contact points slip away under a load. Stepped-on foliage can break and rocks can roll away, robbing a biped of its precious contact forces. In these cases, the previously grounded foot must quickly find a new contact point to facilitate balance, resulting in fast ground impacts. This work presents a force-based controller to balance under these dynamic terrain effects, and demonstrates it on the bipedal robotic platform, "Tallahassee Cassie," (sometimes referred to as "Cassie") as depicted in Fig. 2.1. Additionally, we show the degree to which stable control is feasible using limited information about the dynamic model of the robot.

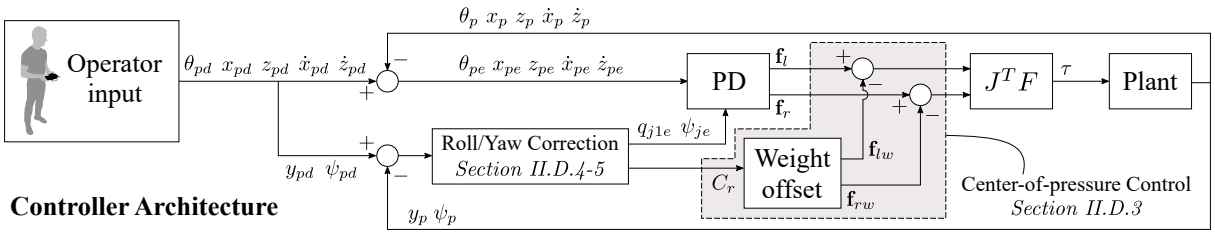

Figure 2.2: Overview of control structure detailed in Section II.

The conceptual pillar of bipedal balancing is the Zero-Moment Point (ZMP) [89] and its popular embodiment in Preview Control [47]. By guaranteeing that the ZMP resides within a robot's support polygon, the robot will not tip over. These core foundations have been greatly extended in the ensuing decades to handle a variety of uneven surfaces and partial footholds [96]. A next-generation approach to humanoid locomotion planned and controlled its joint motions using the concept of centroidal momentum. Some DARPA Robotics Challenge (DRC) humanoids using centroidal momentum could perform vehicle egress maneuvers while the vehicle was being actively bounced [56] - a form of dynamic terrain. Some DRC methods adjusted for unintended robot compliance by modeling it as a quasistatic error [29].

To extend the capability of humanoids, control methods for these fully-actuated machines have been adapted in various ways to explicitly accommodate deformable contacts. Some of these methods work by modeling the deformation and informing the control accordingly. One mode of contact deformation was demonstrated with soft shoe soles, where a deformation model enabled trajectory generation for walking [81]. The humanoid, DURUS, which bore a soft spring in its foot, was controlled via a full-order model-based optimization of the built-in compliant dynamics [38]. A physics-based modeling approach enabled planar bipedal walking on granular media (using poppy seeds as an experimental sand proxy) by modeling and exploiting measured properties of granular materials [102], and some work even uses machine learning to estimate the properties of ground-contact in-situ [15],[74],[18].

Other fully-actuated methods handle deformable contact through various forms of force/torque control for which the terrain properties need not be modeled. A ZMP approach on soft terrain, implemented on WABIAN-2R [49], uses an estimate of the robot's foot roll and torque control to correct its balance as the ground deforms. Taking a more general force/torque control approach, NASA's Valkyrie is built with series compliance to enable torque control [63], and DLR's TORO

robot [27] was torque-controlled to balance on soft gymnasium mats [35], showing robustness to soft terrain. Each of these approaches use highly geared motors and torque sensors to enable force control, limiting their ability to handle dynamic impacts.

Some researchers are achieving force-controlled locomotion with more dynamic impacts by using hardware without the full actuation typical of humanoids. The ongoing work through the HUME and Mercury series point-foot bipeds has shown balancing on uneven surfaces in the sagittal plane [53] and the use of stepping to balance in 3D [52]. They do so using a whole-body variant of operational space control (OSC) [51] which uses a full-body dynamic model of the robot to control task-space forces. Other legged platforms such as quadrupeds have implemented force control methods that functioned through the dynamic impacts of trotting [24] and slippery surfaces [43].

Finally, multiple control methods have been applied to control the Cassie series of robots from Agility Robotics (to which Tallahassee Cassie belongs). Firstly, Cassie's predecessor robot, ATRIAS/MARLO, was controlled to walk in a variety of soft outdoor terrains using control based on reduced-order models [40] and a combined hybrid-nonlinear/machine-learned policy [23]. Methods for controlling Cassie robots to walk include deep reinforcement policy learning [100] and feedback control based on reduced-order models [101]. Using control outputs that are kinematically defined, "Cassie Blue" was able to stand on uneven surfaces and walk on a variety of outdoor terrains [33]. The Cassie-series robot is notable for sporting series-springs and low-friction cycloidal drives - suitable for high-fidelity force control subject to dynamic impacts, which this work aims to exploit.

2.3 Methods

2.3.1 Conventions

All quantities in the body coordinate system have a left subscript $\mathbf{B}$, and quantities without subscript are in the world coordinate system. Vectors can be identified as bold, upright, and lowercase letters $\mathbf{p}$ and matrices are bold, upright, and uppercase $\mathbf{K}$. Scalar values are italicized.

2.3.2 Controller Design

We treat balancing as positioning and orienting the pelvis without regard for the contact positions and achieve it with Fig. 2.2. This method assumes the legs are massless, and are used only as a means to produce desired forces. This design results in a controller that is not concerned with maintaining a specific foot placement, and instead administers necessary motor torques to achieve desired pelvis orientations and positions regardless of foot placement as seen in Fig. 2.3. We use

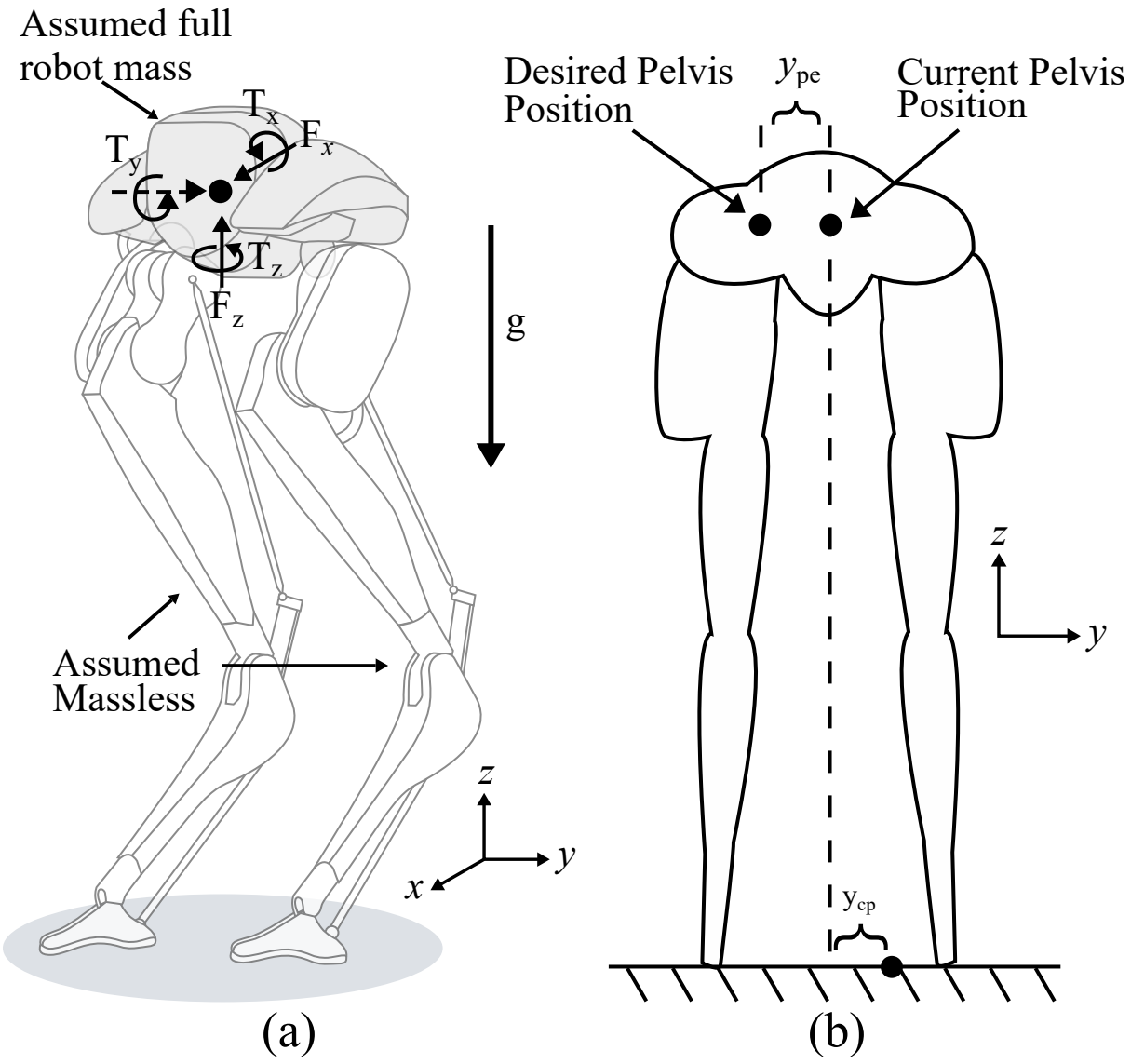

Figure 2.3: (a) Cassie is controlled by commanding the listed forces and torques on the pelvis. The pelvis is assumed to be the only component with mass. (b) Cassie can regulate the y position of its pelvis by controlling the desired center of pressure.

a traditional PD controller to command desired forces to the pelvis. The resulting desired forces are mapped in Cartesian space and converted to motor torques through the multiplication of the Jacobian transpose. We denote the current angles and positions in the world frame as

$$\mathbf{p}_i = [\phi_i \quad \theta_i \quad \psi_i \quad x_i \quad y_i \quad z_i]^T \tag{2.1}$$

where ϕ_i, θ_i, and ψ_i are angles about the x, y, and z axe, x_i, y_i, and z_i are the Cartesian positions, and subscripts $i \in \{p \quad r \quad l\}$ are for the pelvis, right foot, and left foot. The desired angles and positions are

$$\mathbf{p}_{id} = [\phi_{id} \quad \theta_{id} \quad \psi_{id} \quad x_{id} \quad y_{id} \quad z_{id}]^T \tag{2.2}$$

and are represented in the same order as current angles and positions. Subsequently, we represent the current and desired velocities using $\mathbf{v}_i$ and $\mathbf{v}_{id}$ and calculate the position and velocity errors with $\mathbf{p}_{ie} = \mathbf{p}_i - \mathbf{p}_{id}$ and $\mathbf{v}_{ie} = \mathbf{v}_i - \mathbf{v}_{id}$. Next, we develop our gain matrices in Eq. (2.3), where subscript $n \in \{P \quad D\}$ denotes proportional and derivative gains.

$$\mathbf{K}_n = \begin{bmatrix} K_{n\phi} & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{n\theta} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{n\psi} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{nx} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_{ny} & 0 \\ 0 & 0 & 0 & 0 & 0 & K_{nz} \end{bmatrix} \tag{2.3}$$

The hand-tuned gains are multiplied by the errors to develop a simple PD controller. The pelvis errors are used to create desired forces for both legs to apply to the ground, so we compute two force vectors. The general equation form is

$$\mathbf{f}_j = \mathbf{K}_P \mathbf{p}_{je} + \mathbf{K}_D \mathbf{v}_{je} \tag{2.4}$$

where $\mathbf{f}_j$ are the torques and forces applied to the pelvis, $[T_x \quad T_y \quad T_z \quad F_x \quad 0 \quad F_z]^T$, from leg $j \in \{l \quad r\}$ as visualized in Fig. 2.3a. When a j subscript is present, it applies to only the left or right leg, whereas the i subscript can refer to the pelvis, left leg, and right leg. Note, F_y is omitted purposefully with Cassie, where the duty of regulating y_p is allocated to a later-described center-of-pressure controller (illustrated in Fig. 2.3b). In Eq. (2.5) we develop the individual Jacobian matrix for each leg.

$${}_B\mathbf{J}_\mathbb{j} = \begin{bmatrix} \frac{\partial({}_Bp_{j1})}{\partial(q_{j1})} & \cdots & \frac{\partial({}_Bp_{j1})}{\partial(q_{j5})} \\ \vdots & \ddots & \vdots \\ \frac{\partial({}_Bp_{j6})}{\partial(q_{j1})} & \cdots & \frac{\partial({}_Bp_{j6})}{\partial(q_{j5})} \end{bmatrix} \tag{2.5}$$

Here, $[q_{i1} \dots q_{i5}]$ represents motor angles, and $[{}_B p_{i1} \dots {}_B p_{i6}]^T$ represents the respective foot angles and positions in the pelvis frame (shown in Fig. 2.4b). The Jacobian transposes are multiplied by pelvis forces, f_j, as seen in Eq. (2.6) to transform individual motor torques. This is done for both the right and left legs independently.

$$\boldsymbol{\tau}_j = [\tau_1 ... \tau_5]^T = {}_B\mathbf{J}_j^T \mathbf{f}_j \tag{2.6}$$

Section II further details specific differences between the formulation of the force vectors and Fig. 2.2 is a representative visualization of the overall control scheme.

2.3.3 "Tallahassee Cassie"

Tallahassee Cassie is a Cassie series robot designed and built by Agility Robotics and inspired by natural biomechanics, as illustrated in Fig. 2.4. Cassie stands about 1 m tall and weighs approximately 30 kg, with each leg being about 10 kg. Each leg contains 5 motors and 2 mechanical springs. A single onboard battery provides power and allows Cassie to operate untethered and outside in unstructured environments. Cassie is operated using a radio controller, which delivers high-level user instructions. The onboard computer then uses MATLAB and Simulink to control Cassie at a rate of 2 kHz. Cassie senses the environment with a limited array of sensors. These sensors include an Inertial Measurement Unit (IMU) and joint encoders, but Cassie uses no vision nor global positioning. The in-series springs dissipate impact forces and prevent damage to the hardware, but can also be used to measure contact forces via deflection. Of the five motors on each leg, four contain cycloidal transmissions, which permit high-gear reduction with low friction and no backlash. This low friction enables accurate transmission of torques, which at a high control rate enables successful force control.

2.3.4 Controller Implementation

We make the following key assumptions to develop the controller. First, the low-friction cycloidal drives allow us to assume that commanded torques are equivalent to the applied torques at the joint. For our tests, the centroid was used as the desired pelvis location as seen in Eq. (2.7). By centering the pelvis in the world frame, the controller can resist external forces up to a limit dependent upon the ground slope, damping coefficient, and foot contact friction. The height of the

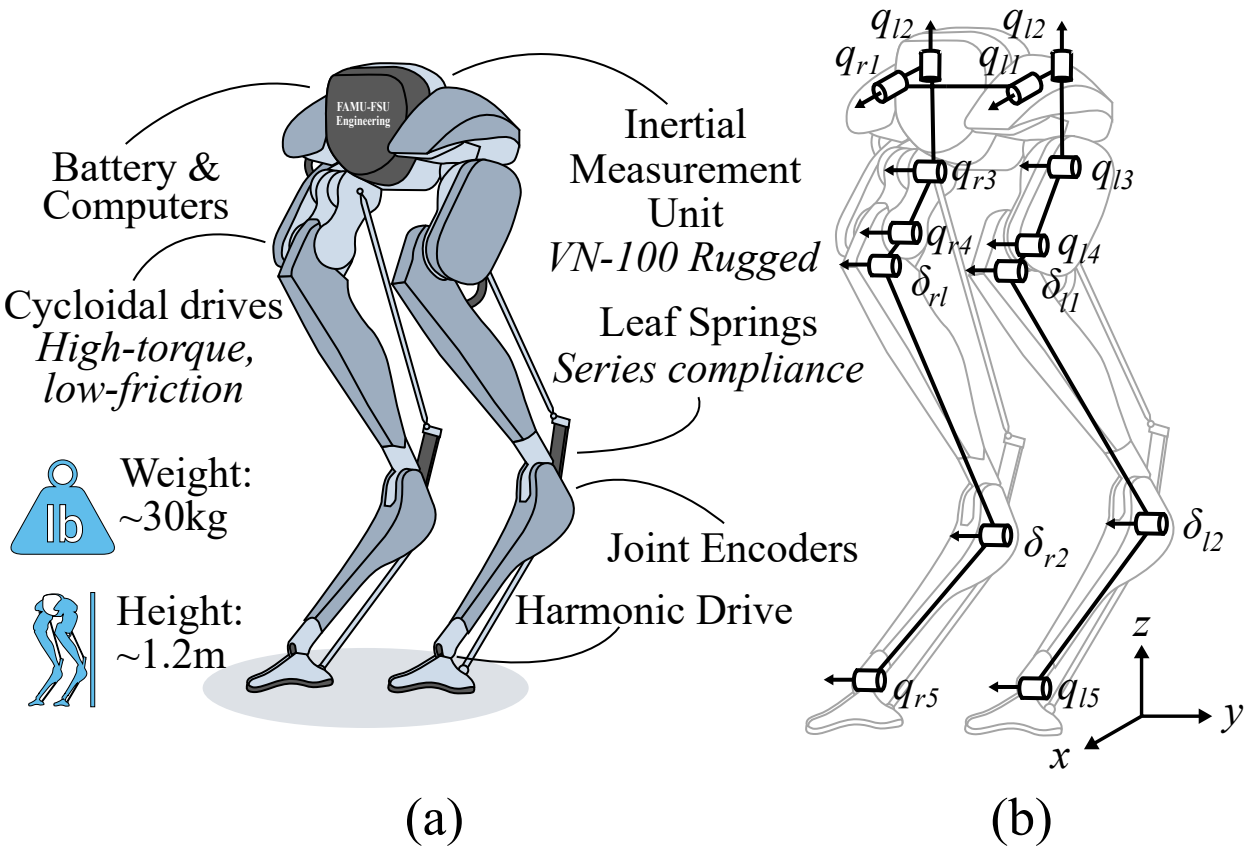

Figure 2.4: (a) Overview of Cassie's hardware and basic features. (b) Kinematic diagram of Cassie showing motor positions and spring deflections.

pelvis is defined by the foot with the largest z distance as can be seen below.

$$\mathbf{p}_p = \begin{bmatrix} \phi_p \\ \theta_p \\ \psi_p \\ x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} \phi_p \\ \theta_p \\ \psi_p \\ \bar{x}_j \\ \bar{y}_j \\ max(z_j) \end{bmatrix} \tag{2.7}$$

To determine the angular orientation of the pelvis, or the first three values of Eq. (2.7), from IMU measurements. To measure the full system state, we use all fourteen encoders (seven per leg), and the IMU. After developing a symbolic expression of the forward kinematics using the MATLAB symbolic toolbox, we found the positions and angles of each foot in the world frame. We then differentiate the feet kinematics and generate a fast-solving Jacobian using the `matlabFunction` command.

Pelvis Velocity Measurement. The pelvis angular velocities are derived directly from IMU measurements. To determine the linear velocity, we used kinematic estimates from both legs, combining their estimates with an average weighted by leg-spring deflections (a rough gauge of contact force magnitude). If both legs had spring deflections from contact, the pelvis' linear velocity was calculated from both feet using a weighted average based on the overall deflection,

$$\mathbf{v}_p = -\frac{\sum v_j(\delta_{j1} + \delta_{j2})}{\sum(\delta_{j1} + \delta_{j2})}. \tag{2.8}$$

After differentiating the forward kinematics, the computed velocity signal had problematic noise which led to motor oscillations from the derivative control terms. We apply a second-order low-pass Butterworth filter with a cutoff frequency of 10 Hz to both the linear and angular velocity derivative gain terms. To further reduce oscillation, a dead band filter to the velocity terms with a range of ± 0.10 m/s was added. After implementation, we noticed that the angular velocity gains did not contribute much to the stability, but vastly increased the probability of inducing system oscillations. This observation motivated us to set the angular velocity $\mathbf{K}_D$ gains to zero. Future work will employ an extended Kalman filter and re-implementation of these $\mathbf{K}_D$ gains.

Calculating Errors. The user sets the desired positions and velocities of the pelvis in the control, resulting in an error between the current and desired state. The system is treated similarly to a rigid body where errors at the feet are opposite of the pelvis errors. With the few exceptions discussed below, these errors are taken and multiplied by the $\mathbf{K}_P$ and $\mathbf{K}_D$ gains to compute pelvis forces.

Center-of-pressure Control. The Cassie series robot uses five motors to control the available six Cartesian degrees of freedom per foot. We found directly controlling all Cartesian forces except for the y force to be an effective control scheme. Instead, the y pelvis error is used to compute a desired center of pressure (y_{cp}) between the two feet as illustrated in Fig. 2.3b and in

$$y_{cp} = y_p - K_{cp} y_{pe} \tag{2.9}$$

where K_{cp} represents a new center-of-pressure gain. Using a simple statics equation we next develop a ratio that determines the necessary weight distribution per leg to achieve the desired y_{cp},

$$C_r = \frac{y_{cp} - y_r}{\sum y_j}. \tag{2.10}$$

When multiplied by the weight, this ratio (C_r) not only ensures the weight is always supported by the controller but also is used to apportion vertical forces to each leg as seen in (2.11) and (2.12)

$$F_{rz} = WC_r \tag{2.11}$$

$$F_{lz} = W - F_{rz} \tag{2.12}$$

$$\mathbf{f}_{jw} = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad F_{jz}]^T \tag{2.13}$$

where W is an estimate of the total weight of the robot. This process is visually described in Fig. 2.3b.

Modifying x Rotation. From the center of the pelvis point to a point internal to the hip motor is a fixed distance, y_h, of 0.135 m. Then we determine whether the legs are inside or outside the form factor of the pelvis using Eq. (2.14). A $y_f > 0$ indicates the foot span is greater than the distance between the left and right hip points.

$$y_f = y_h - \frac{\sum \mid y_j \mid}{2} \tag{2.14}$$

Eq. (2.15) uses the y_f to determine the q_{i1} motor angles (reference Fig. 2.4b) necessary to induce zero roll when the pelvis is centered. These two motors are the only two directly associated with control of the pelvis roll.

$$\mathbf{q}_{j1cd} = \tan^{-1}\left(\frac{y_f}{z_j}\right) \tag{2.15}$$

Eq. (2.16) calculates the additional angles necessary for when y_d is not directly between the feet.

$$\mathbf{q}_{j1yd} = \tan^{-1}\left(\frac{y_{pd}}{z_j}\right) \tag{2.16}$$

Finally, we sum (2.15) and (2.16) to get the desired hip motor positions.

$$\mathbf{q}_{j1d} = \mathbf{q}_{j1yd} + \mathbf{q}_{j1cd} \tag{2.17}$$

The desired hip motor angles to induce zero roll were developed using the kinematic equations and IMU data. After testing, this method of control proved to be more effective than directly controlling the pelvis roll based on the IMU measurement alone.

Modified z rotation. To keep the feet from bowing in or out (yaw rotation), individual foot rotation control around the z axis is included. To simplify the controller, we set $\psi_{ld} = \psi_{rd} = \psi_{pd}$ and the feet errors are calculated in the same manner previously mentioned,

$$\boldsymbol{\psi}_{je} = \psi_{jd} - \psi_j. \tag{2.18}$$

Forces. Using the modified error calculations we compute the force vectors for the left and right foot separately. Note that the previous modifications now result in

$$\mathbf{p}_{je} = [\mathbf{q}_{j1e} \quad \theta_{pe} \quad \boldsymbol{\psi}_{je} \quad x_{pe} \quad 0 \quad z_{pe}]^T \tag{2.19}$$

and

$$\mathbf{v}_{je} = [0 \quad 0 \quad 0 \quad \dot{x}_{pe} \quad 0 \quad \dot{z}_{pe}]^T. \tag{2.20}$$

Multiplying errors by the gains and subtracting the center of pressure force yields

$$\mathbf{f}_j = \mathbf{K}_P \mathbf{p}_{je} + \mathbf{K}_D \mathbf{v}_{je} - \mathbf{f}_{jw} \tag{2.21}$$

The $\mathbf{K}_P$ and $\mathbf{K}_D$ gains are identical for both legs. Using Eq. 2.6, we find the torques necessary to drive the pelvis to the desired position. The computed torques are commanded and realized using Cassie's onboard motor controllers.

2.3.5 Multibody Simulation

To test how the controller performs in simulation, we have a multibody model of Cassie built in Simulink Multibody in MATLAB 2017b. This platform provides us with a 3D simulation that accounts for impact forces, inertia, and many other nuances present in Cassie's multibody dynamics. The overall model has the same number of joints, motors, and springs and similar masses, kinematics, and moments of inertia as estimated on Cassie. To reduce the simulation computation time the contact model utilizes spheres at the front and back of each toe for a total of four points of contact including both legs.

2.4 Experimental Setup

During each trial, Cassie is kept on a safety harness to prevent damage in the event of failure (Fig. 2.5). This harness is kept slack throughout each experiment to minimize interference. Cassie is then booted and calibrated, after which power is supplied to the motors, and motor torques are slowly ramped up as Cassie comes to a complete standing pose with assistance. After Cassie is standing, perturbations are applied. Then, power is cut to the motors and data stops being recorded from the onboard computer. During each trial, Cassie's onboard computer logs data at a rate of 2 kHz.

Four types of balance perturbations were applied to Cassie. The "Leg Spread" is conducted by placing a board under Cassie's foot that is used to slide Cassie's foot right and left, simulating unstable terrain. This board is also used for the "Lift and Lower" where the board is slowly lifted, causing Cassie's foot to lift and be on uneven footing, which constitutes uneven terrain. To simulate soft terrain in the "Soft Foam and Push," one of Cassie's feet is placed on foam mats and Cassie is given slight pushes. Then, to simulate losing a foothold in the "Box Drop," Cassie is placed with one foot on a box, and this box is quickly forced out from underneath the foot, causing that foot to fall to the ground.

2.5 Results

Each of the four perturbation scenarios is shown visually in Fig. 3.5 and their respective pelvis trajectories are plotted in Fig. 2.7. Tallahassee Cassie was able to remain balanced while (1) having its leg pulled to the side 55cm, (2) having its leg lifted vertically by 30cm, (3) standing on soft foam and pushed, and (4) when a 17cm-tall box is pulled from under its foot. In each case, the controller accommodates the disturbance and settles back near the desired pelvis location. These tests experimentally demonstrate the effectiveness of the simple force controller despite highly simplified assumptions.

2.6 Conclusions

The presented force-controlled balancing algorithm allowed Tallahassee Cassie to balance on uneven, moving, and soft surfaces - which we call dynamic terrain. Even when a contact point was swiftly removed by pulling out a supporting box, the force controller quickly stomped the leg

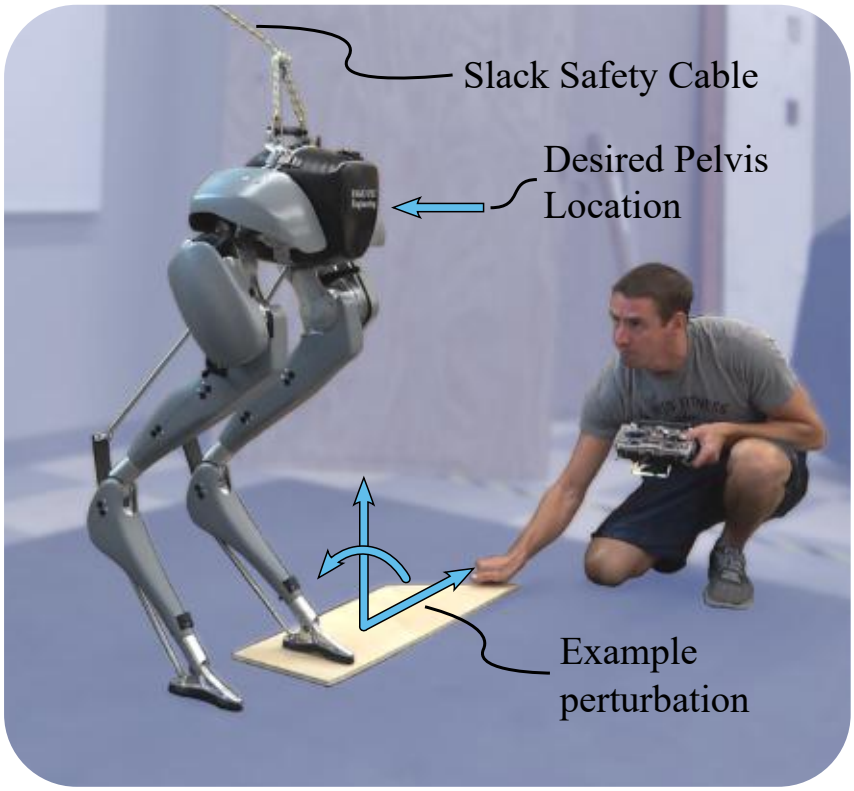

Figure 2.5: Experimental setup for testing the balancing controller on Tallahassee Cassie for each of four balance perturbations.

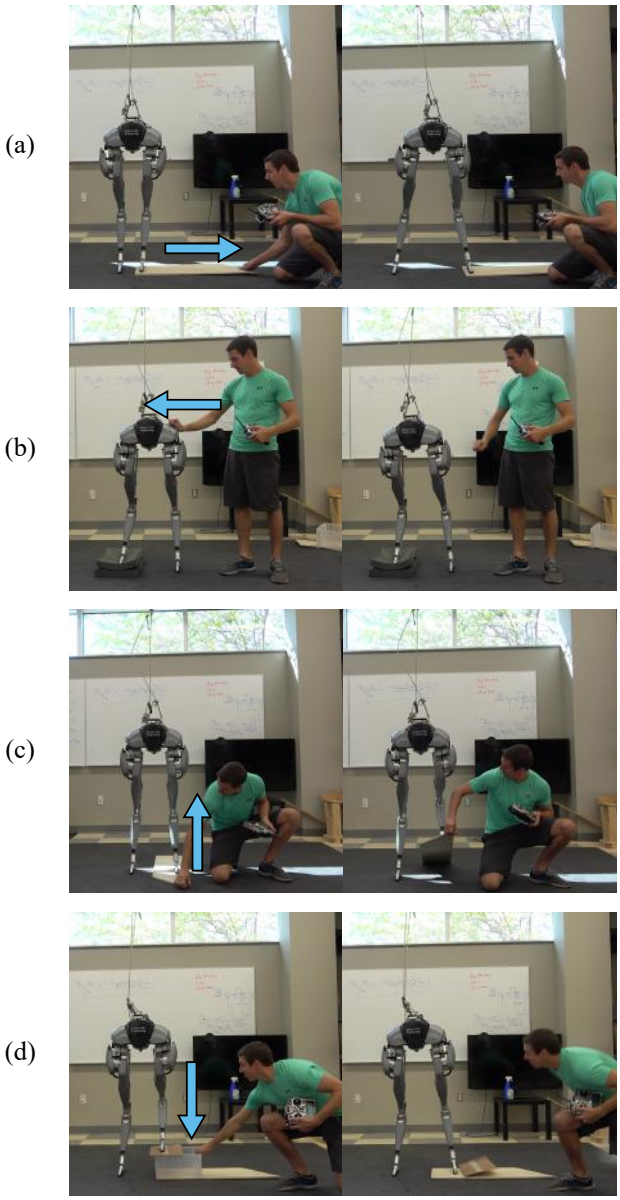

Figure 2.6: Examples of different perturbations applied on Tallahassee Cassie. (a) "Leg Spread": Cassie's foot is moved by sliding a board. (b) "Soft Foam and Push": One of Cassie's feet is supported by soft foam and the pelvis is pushed. (c) "Lift and Lower": Cassie's foot is raised and lowered using a board. (d) "Box Drop": A box is quickly pulled from under Cassie's foot, forcing it to recover from a fall.

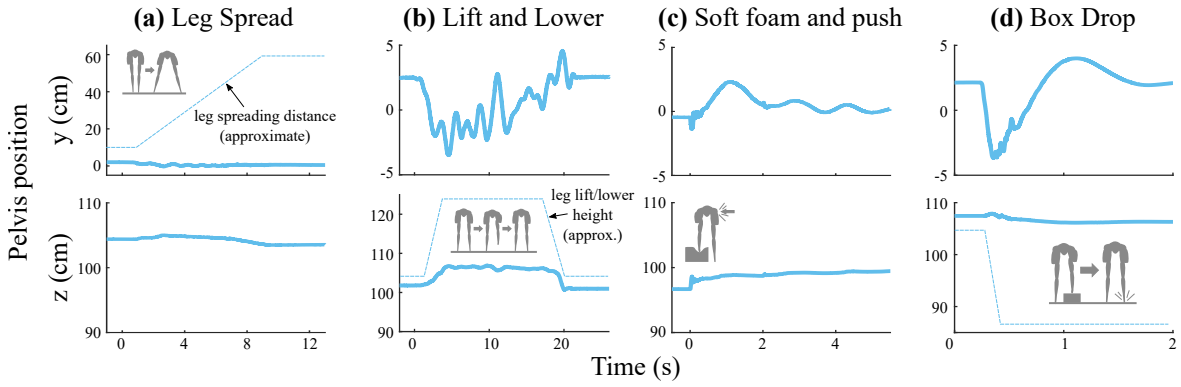

Figure 2.7: Experimental data from each of four perturbation tests. (a) Cassie's foot is pulled away while maintaining a stable height. (b) Cassie's foot is moved up and down and Cassie returns to its initial position. (c) Cassie is pushed on soft foam and returns to a stable position. (d) Cassie's foot falls rapidly as a box is pulled out from beneath its foot, and Cassie recovers to its initial position. Solid lines are measured pelvis data, and dashed lines represent approximated leg perturbations (*e.g.*, height of leg lift). Note: y-positions are measured as the distance from the desired centered position. See supplementary video for experimental footage.

to a new contact point and recovered. The controller is realizable in part because of its hardware platform. Tallahassee Cassie is equipped with low-friction cycloidal drive transmissions (or transparency), and thus sufficiently high-fidelity torque control is possible. Further, Cassie's series springs protect the structure from impacts, allowing the controller to quickly and safely move to new contact points when the ground is pulled away.

We further note that this controller functioned despite the vastly simplifying assumptions made to implement the controller. Cassie has soft and passive series-elasticity in four joints, and significant mass and inertia in its long legs (especially compared to its compact torso). Yet, the total robot mass is assumed to be completely contained in the pelvis with the legs being massless, and the spring deflections are excluded from the kinematics computations. Additionally, there is no explicit modeling or control logic for swing phases, so any airborne behavior for a leg strictly emerges from the force controller. These control results are a foundation for further control on real-world terrain, including walking and running on dynamic terrain. Immediate next steps will seek to incorporate stepping logic to broaden the disturbance capability of the robot.

CHAPTER 3

THROUGH-CONTACT MOTION PLANNING

The vast majority of legged robot controllers use a predefined contact sequence with clock-based gait timing. Although effective, objectives that require immediate action (such as a change in desired states or avoiding a moving obstacle) may not be viable due to these predefined limits. Here we present a controller that allows for instantaneous gait modifications by leaving the contact time and sequence as free variables for single-legged models. The generated motion plans assume a linear approximation of the model, which enables convex optimization methods to solve these plans in milliseconds. This section is a preprint of a pending conference paper named "Through-Contact Monopod Motion Generation via MPC: Convex Formulation and Locomotion Analysis" submitted to the 2024 International Conference on Intelligent Robots and Systems.
Specific contributions of this chapter include:

1. A real-time through-contact motion planner that uses the model prediction to iteratively update a mode schedule prediction
2. Planar simulations showing the receding horizon planner's ability to achieve stable limit cycles without the use of periodicity constraints
3. Demonstration of standing/hopping transitions and deadbeat (single gait cycle) disturbance rejection
4. Numerical experiments showing the relative insensitivity of the gait dynamics to the choice of MPC time horizon.

3.1 Abstract

Here we present a convex motion planner for single-legged robots that generates behaviors in real-time without using predefined contact sequences or mode durations. Similar to existing model-predictive controllers (MPC), this method iteratively solves quadratic programs (QPs) to generate trajectories for linearized plant models over a receding planning horizon. However, the presented method also iteratively updates the predicted contact sequence with each control cycle. This allows for through-contact motion planning without smoothing contact conditions or including

any nonlinear constraints. Despite the contact predictions being locally and iteratively updated, planar simulations of the controller exhibit a wide range of monopod behaviors. By only setting a desired mean speed and mean height in the optimization cost, the monopod hops with a stable limit cycle, transitions between hopping and standing within one gait cycle, and demonstrates deadbeat disturbance rejection. Analysis of the exhibited gait cycles indicates that key markers such as gait period are highly dependent on the dynamics and task, not controller hyperparameters such as the time horizon. This suggests that the design choices that render the through-contact MPC fast-solving are not impeding behaviors that are well suited to the robot's dynamics and task.

3.2 Introduction

Hopping and jumping are critical locomotion modes for highly agile animals, from parkour-performing humans to arboreal monkeys to leaping goats, and have served as inspiration for legged robots for decades [72, 34]. However, leaping between contact and no-contact modes induces hybrid dynamics and underactuation which have been historical challenges for control. In the late-2010s, real-time motion planning (*e.g.*, MPC) emerged as a powerful tool for controlling legged robots in agile locomotion regimes (*e.g.*, trotting quadrupeds [24], jumping humanoids [42]). While these methods generate more robust locomotion to disturbances, they assume specific contact sequences and timings *a priori* to speed up their computation – this limits the range of extemporaneous behaviors they can exhibit. This section presents a real-time MPC approach for monopod hoppers that plans through-contact, assuming no contact sequence or timings *a priori*.

Controllers and heuristics for monopods date back to the Raibert controllers [72]. These controllers specifically regulated gait quantities such as energy [2] to achieve bounding at a desired height and speed but required manual gain tuning. Later methods specifically developed control policies for canonical models, such as the spring-loaded inverted pendulum (SLIP) [78] (and variations thereof) for stable [69, 5], robust [54, 88], and deadbeat control [97]. However, designing these policies requires simulations that span the state space in a brute-force manner, making policy adjustments impractical during operation. Other approaches use feedback linearization to allow for on-the-fly planning computations with specific dynamic hopping models [66].

Optimization-based legged control methods offer more flexibility with respect to plant models. Solving trajectory optimizations offline generates fast and stable hopping [31] when periodic gaits are enforced via constraints [98] or given sufficiently long multi-step time horizons [39]. However,

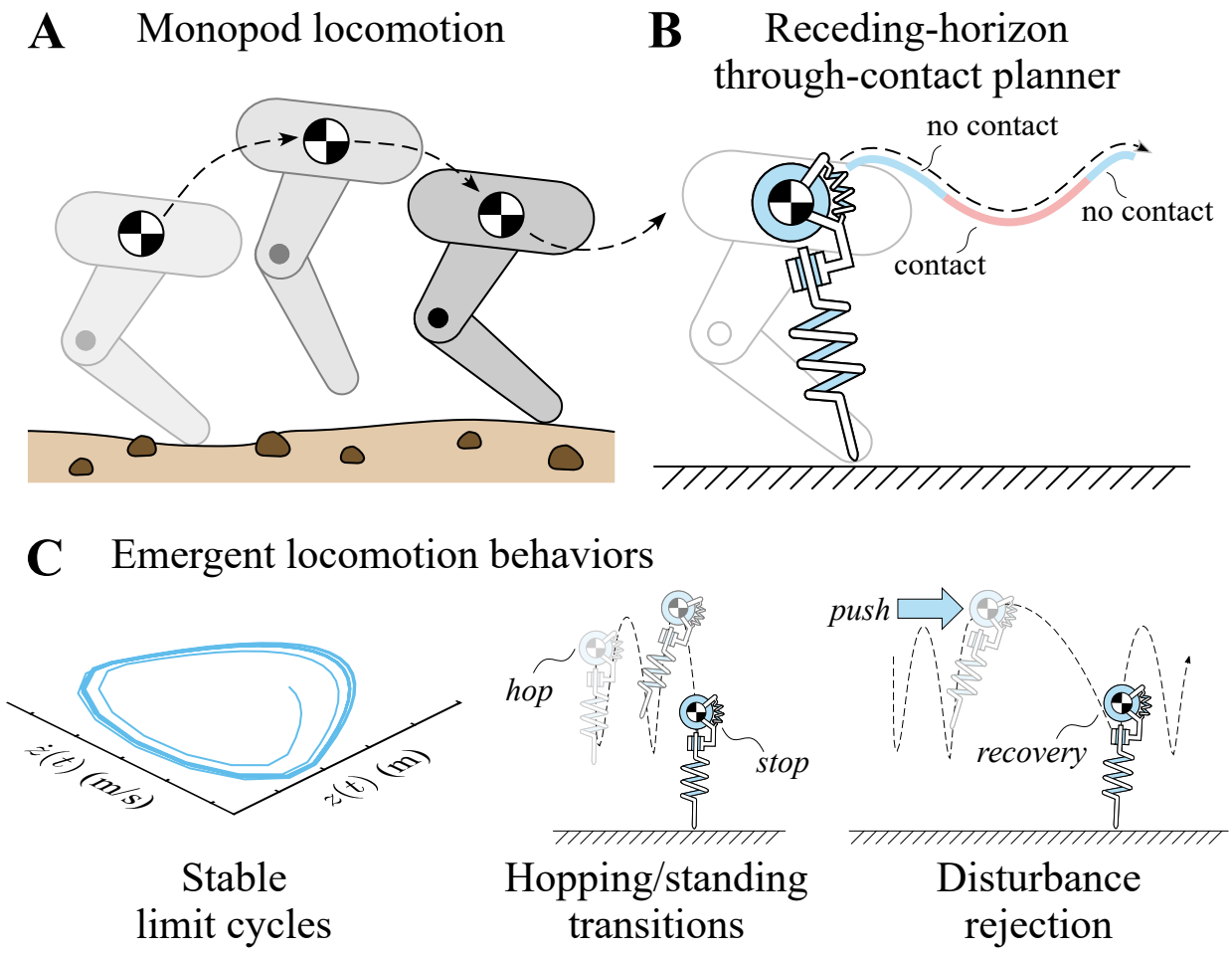

Figure 3.1: **A.** An illustration of monopod locomotion. **B.** This work presents a through-contact planner for the control of monopods. **C.** The controller yields canonical legged behaviors such as (1) stable limit cycles, (2) hopping/standing transitions, and (3) disturbance rejection.

these behaviors are limited to the trajectories generated before operation. In contrast, model-predictive control (MPC) solves these trajectory optimizations in real-time with a short and receding time horizon. MPC is responsible for highly dynamic maneuvers for many legged robot morphologies [24, 80]. However, to achieve the fast computation speeds required for real-time operation, contact sequences and mode duration (*i.e.*, footfall timing) are assumed *a priori*.

Through-contact and contact-implicit optimizations offer the freedom to change contact sequence and timing on-the-fly. A notable example is the use of linear complementarity problems (or LCPs) to express contact as a constraint relating distance-to-contact to contact force [68]. Recent work has accelerated the solve time of complementarity-constrained trajectory optimizations [22, 8] to 15-30Hz, but methods that rely on convex quadratic optimizations are considered faster and the solvers more robust [84]. Differential dynamic programming methods can achieve similar solve times to LCP methods by using invertible contact models [19]. Smoothing the contact model can accelerate through-contact motion planning [62] at some cost to dynamical accuracy while globally optimal contact plans can be found using convex (but-slower) mixed-integer approaches [25]. While quickly and reliably solving the general through-contact motion planning problem is an ongoing challenge, we posit that the dynamics of monopod locomotion are a special case that can be effectively planned through-contact with a simpler and faster algorithm.

3.3 Methods

3.3.1 Conventions

The following notation is used throughout the chapter: R, R^n, $R^{n\times m}$ represent the space of real numbers, vectors with length n, and matrices with n rows and m columns. Scalar values are lowercase and italicized (*e.g.*, $x \in R$); vectors are lowercase and bold (*e.g.*, $\mathbf{q} \in R^n$); and matrices are uppercase and bold (*e.g.*, $\mathbf{A} \in R^{n\times m}$). Variables $\mathbf{p}$, $\mathbf{v}$, $\mathbf{u}$ are positions, velocities, and control inputs respectively. Finally we define the robot state vector as $\mathbf{q} = [\mathbf{p},\ \mathbf{v}]^T$ and its derivative $\dot{\mathbf{q}} = [\dot{\mathbf{p}},\ \dot{\mathbf{v}}]^T$. For these variables and others in the following sections, notations represent a single node in the optimization unless explicitly stated with a k. Similarly, all variables presented are time-varying except for the model's physical properties mass m, gravity g, spring stiffness $[k_1, k_2]$, nominal lengths $[l_o, \theta_o]$, and maximum length l_{max}.

3.3.2 Assumptions

For the models used here, we make some general simplifying assumptions. First, we assume the actuated spring model leg is massless. During stance, we also assume the actuated spring model's ground contact point is fixed. Finally, the model predictions are integrated using a trapezoidal scheme but the controller is simulated using MATLAB's ode45 variable-time-step differential equation solver.

3.3.3 Through-Contact Planning

Here we discuss our approach which allows us to formulate a through-contact motion planning optimization as a convex quadratic program (QP). It is well established that linear dynamic models of legged systems when given a constant contact sequence and mode duration, can have their motion plans generated using convex quadratic programs [32]. Our method leverages this fact by predicting a mode schedule, assuming the constant mode schedule for a single optimization, and then iteratively updating the predicted schedule between MPC optimizations. The steps are as follows and illustrated in Fig. 3.2.

1. **Initialize:** Generate a guess of the planned trajectory for a fixed time horizon.
2. **Generate Modes:** Check contact guards along the planned trajectory to generate a mode schedule guess
3. **Linearize:** Linearize the plant dynamics along the trajectory given the current mode schedule
4. **Optimize:** Formulate and solve the motion planning problem over the time horizon, T, as a convex quadratic program
5. **Execute and Resample:** Enact the motion plan for sampling time, T_s, and sample the current state as the new initial condition

Utilizing an estimated planned trajectory allows us to both estimate a contact sequence and develop an accurate linearization of the dynamics. The first iteration assumes the system is stationary. Each successive iteration uses the dynamic set associated with the projected mode (stance or flight) to generate a motion plan. Changes in the contact guards are then identified after the motion plan is generated, and the mode schedule is updated accordingly. The contact guards are determined by identifying breaches in the kinematic limits assigned to each node from the dynamics. This approach will update mode changes along the motion plan, and find up to one additional

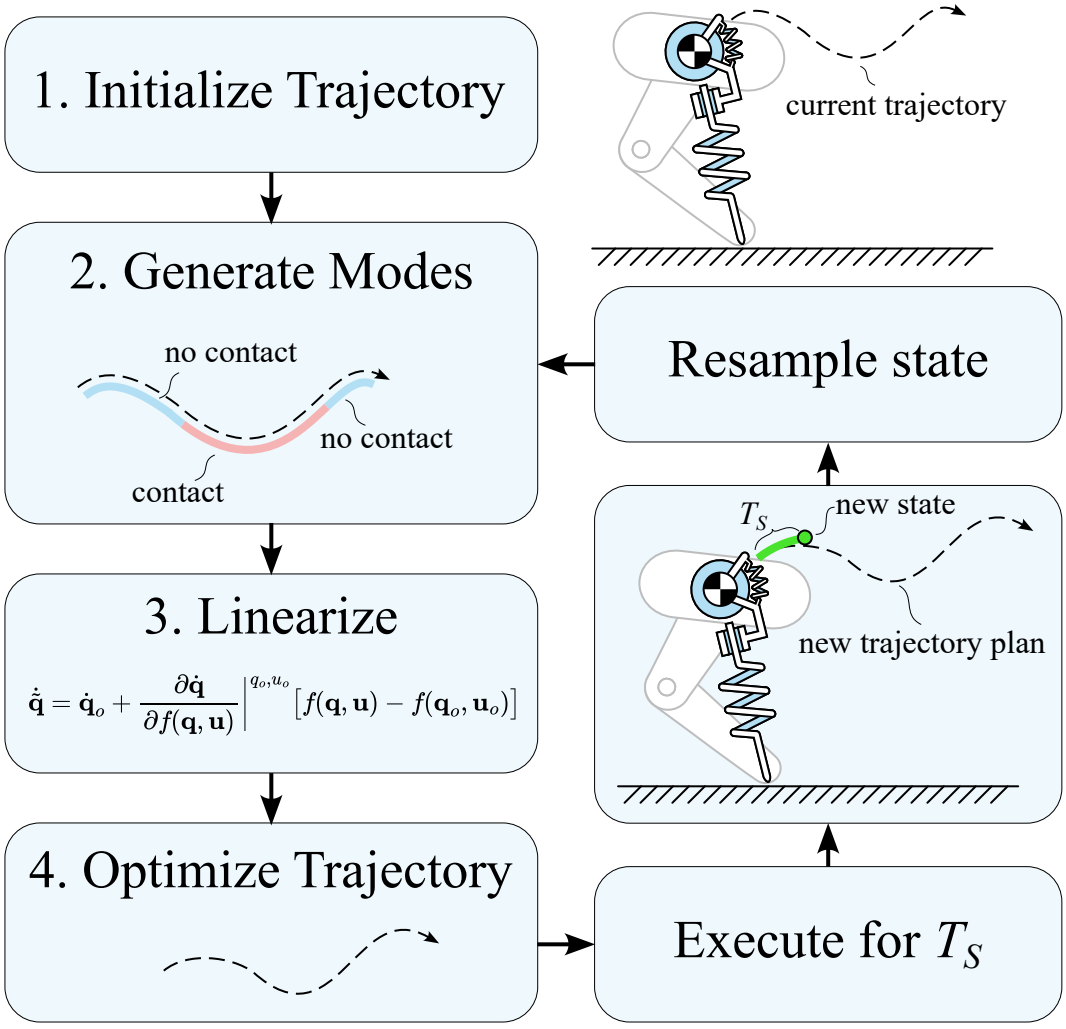

Figure 3.2: Overview of the through-contact model-predictive control procedure.

mode change per iteration. The first few iterations are run without simulating the system's motion to essentially warm-start the optimization and generate a reasonable initial trajectory.

For the nonlinear monopod presented here, we identify when the toe would be above or below ground using the leg length and angle. This allows us to determine which set of dynamics to use for each node in the optimization. These nonlinear dynamics are used to simulate the system but are not suited for use with a convex optimization. After this determination, the states in the previously generated trajectory are used with a first-order Taylor Series to linearize the dynamics.

$$\dot{\tilde{\mathbf{q}}} = \dot{\mathbf{q}}_o + \frac{\partial \dot{\mathbf{q}}}{\partial f(\mathbf{q}, \mathbf{u})}\bigg|^{q_o, u_o} \big[f(\mathbf{q}, \mathbf{u}) - f(\mathbf{q}_o, \mathbf{u}_o)\big] \tag{3.1}$$

where the robot state vector, its derivative, control inputs, and design vector $f(\mathbf{q}, \mathbf{u})$ are evaluated at operating points $\mathbf{q}_o$, $\dot{\mathbf{q}}_o$, $\mathbf{u}_o$ to find a linear form of the dynamics $\dot{\tilde{\mathbf{q}}}$. The approach presented mimics the methods presented in [93], where the operating point values come from the previous optimal trajectory generated by the MPC.

Typically these dynamics are then used with some integration scheme to constrain the nodes to each other (as seen in Sec. 3.3.5). When we encounter mode changes we remove the continuity constraint for the discontinuous states (*i.e.*, the toe position and velocity) and act as though the first state of the phase has its own set of initial conditions. Finally, we optimize the problem and simulate the nonlinear dynamics using the inputs developed from optimization. The results of the simulation then become our initial state, and the process starts over.

3.3.4 Dynamical Models

We demonstrate our through-contact MPC method using two dynamic monopod models in the sagittal plane. We first introduce a point-mass model we dub the **simplest monopod** (SM) model to clearly illustrate the method. Secondly, we introduce an **actuated spring-loaded monopod** (ASLM) model, similar to the spring-loaded inverted pendulum [78] but with rotational and prismatic actuation [88], to demonstrate extensibility to more-complex leg morphologies.

Simplest Monopod Model. The point mass model's state vector has two positions $\mathbf{p} = [x, z]^T$ and velocities $\mathbf{v} = [\dot{x}, \dot{z}]$. The dynamics use simple Cartesian forces and gravity per

$$\dot{\mathbf{v}} = \left[\frac{F_x}{m}, \frac{F_z}{m} - g\right]^T \tag{3.2}$$

and assumes the control inputs $\mathbf{u} = [F_x, F_z]^T$ is zero once the point mass reaches a user-defined vertical height.

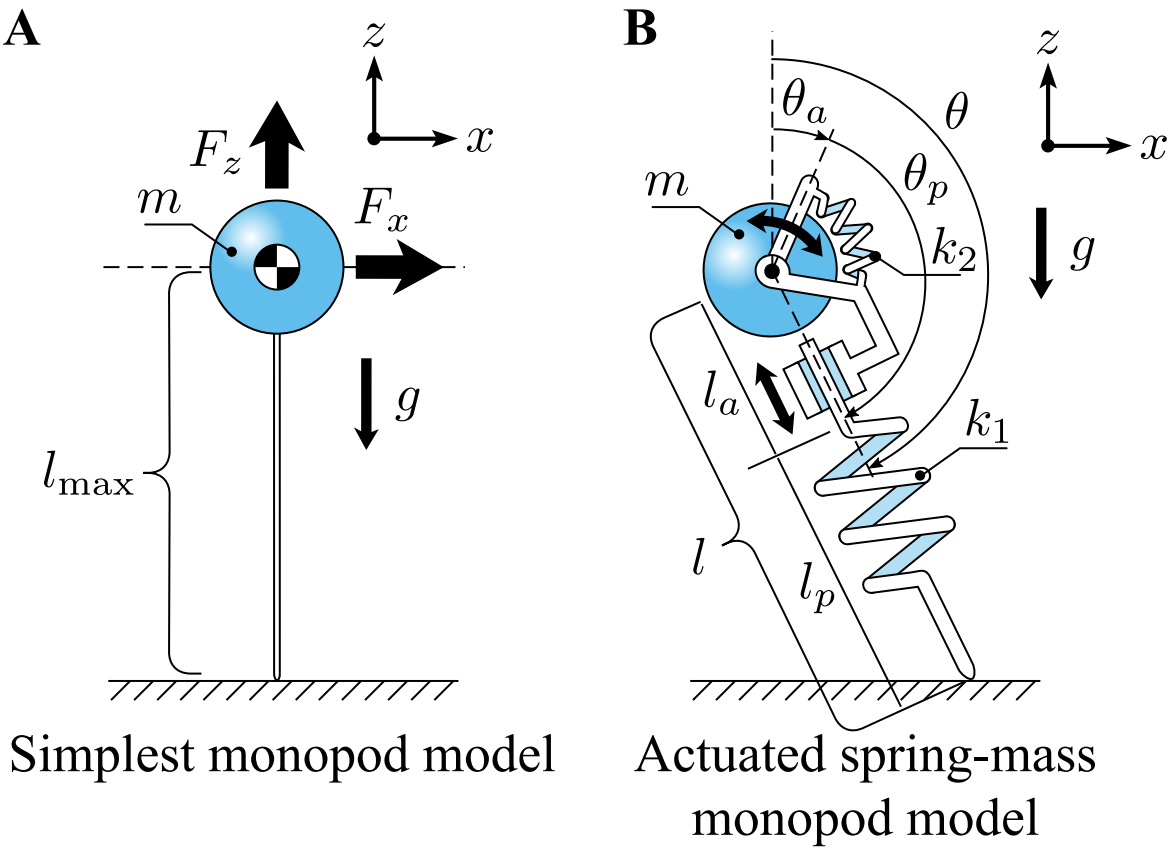

Figure 3.3: Diagram of the two kinematic models used for the optimizations. **A.** The simplest monopod, showing the z based l_{max} limit which dictates what dynamic model to use. **B.** Actuated spring-mass system with both a prismatic and rotational spring. The calculated l determines which set of dynamics to use.

Actuated Spring-loaded Monopod Model. The ASLM model state vector includes body positions $\mathbf{p}_b = [x_b, z_b]$, toe positions $\mathbf{p}_t = [x_t, z_t]$, and the leg actuation length & angle $\mathbf{p}_a = [l_a, \theta_a]$ for combined position vector of $\mathbf{p} = [\mathbf{p}_b, \mathbf{p}_t, \mathbf{p}_l]^T$. This model differs from the traditional "minimalist coordinate" approach but allows us to estimate and simulate when we expect ground contact will occur. We formulate the nonlinear dynamics for each phase (stance and flight) using a Newton-Euler approach. The model has two springs, one prismatic along the leg, and a second rotational between the leg and rotational actuator. Both the leg length (l) and angle (θ) are determined using the actuator positions ($\mathbf{p}_a$) and spring deflections (l_p, θ_p)

$$l = l_a + l_p, \quad \theta = \theta_a + \theta_p \tag{3.3}$$

In flight, the spring's passive and nominal length are equal ($l_p = l_o$ and $\theta_p = \theta_o$), so the toe positions are simply

$$\mathbf{p}_t = \mathbf{p}_b + [l\sin(\theta),\ -l\cos(\theta)]^T. \tag{3.4}$$

During this phase the body follows a simple ballistic trajectory, however, the toe acceleration can vary depending on the prismatic ($\ddot{l}_a$) and rotational ($\ddot{\theta}_a$) actuator accelerations.

$$\dot{\mathbf{v}}_b = [0,\ -g], \quad \dot{\mathbf{v}}_t = \frac{d^2}{dt^2}\mathbf{p}_t, \quad \dot{\mathbf{v}}_a = [\ddot{l}_a,\ \ddot{\theta}_a] \tag{3.5}$$

In stance, the springs deflect so the previous assumptions no longer hold. Instead, the distance between the body and toe is found via $\mathbf{r} = \mathbf{p}_t - \mathbf{p}_b$, which is then used to find the forces from spring deflections.

$$\begin{aligned} F_l &= k_1(l_o + l_a - l) \quad where \quad l = ||\mathbf{r}||_2 \\ F_\theta &= k_2(\theta_a - \theta) \qquad where \quad \theta = \sin^{-1}\left(\frac{r_x}{l}\right) \end{aligned} \tag{3.6}$$

The equations of motion are determined by parsing forces into their respective cartesian directions $\left(e.g.,\ F_{l,x} = F_l\frac{r_x}{l}\right)$.

$$\begin{aligned} \dot{\mathbf{v}}_b &= \left[\frac{1}{m}(F_{\theta,x} - F_{l,x}),\ -\frac{1}{m}(F_{\theta,z} + F_{l,z} + mg)\right], \\ \dot{\mathbf{v}}_t &= [0,\ 0], \qquad\qquad \dot{\mathbf{v}}_a = [\ddot{l}_a,\ \ddot{\theta}_a] \end{aligned} \tag{3.7}$$

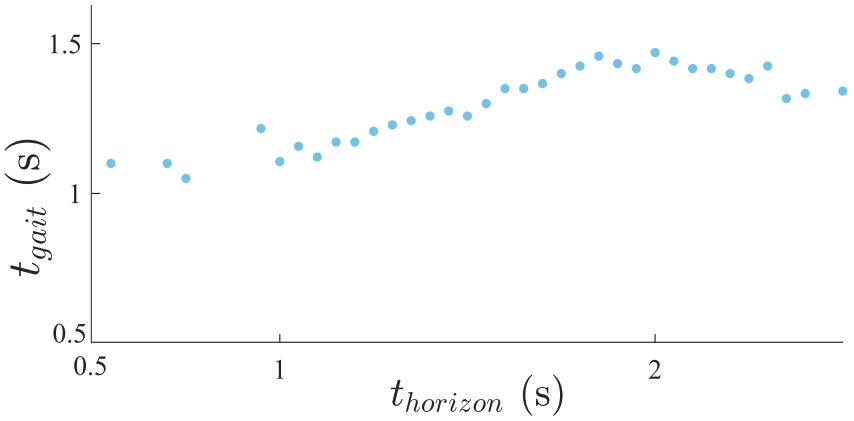

Figure 3.4: Actuated mass-spring model results with a defined MPC time horizon and the corresponding gait cycle time for an unchanged objective

3.3.5 Model Predictive Control

MPC utilizes online optimizations to track desired trajectories, but can also be used for online motion generation with loose (or non-existent) tracking requirements. We use it to generate trajectories (**p,v**) and associated control inputs (**u**). As is typical in MPC, we facilitate model prediction in our equality constraints using a Trapezoidal integration of our system dynamics. At each control loop, we constrain the measured states to equal the initial trajectory states ($\mathbf{q}_1$). The optimization uses a multi-part quadratic cost, and linear constraints per:

$$\begin{aligned} \min_{\mathbf{q}_k, \mathbf{u}_k} \quad & J(\mathbf{q}, \mathbf{u}) \\ \text{s.t.} \quad & \dot{\mathbf{q}}_k = \mathbf{A}\mathbf{q}_k + \mathbf{B}\mathbf{u}_k && \text{(Dynamics)} \\ & \mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k \Delta t && \text{(Model Prediction)} \\ & h(\mathbf{q}_k, \mathbf{u}_k) \leq 0 && \text{(Task Constraints)} \end{aligned}$$

and is detailed in later sections.

3.3.6 Cost Function

The cost function $J(\mathbf{q}, \mathbf{u})$ has a corresponding weighting matrix, $\mathbf{W}$, and is specific to each task and model (J for brevity). The cost function consists of two components, the desired state error $J(\mathbf{q})$ and an actuator acceleration penalty $J(\mathbf{u})$.

$$J(\mathbf{q}) = ||\mathbf{q}_k - \mathbf{q}_{ref}||^2_{\mathbf{W}_{\text{target}}} \qquad J(\mathbf{u}) = ||\mathbf{u}_k||^2_{\mathbf{W}_{\text{actuator}}} \tag{3.8}$$

The spring deflection between the actuator and hopper components is what produces the forces acting on the body. The optimization problem is minimizing the actuator acceleration rather than the forces produced. The ratio of the target to actuator weight matrices is 5000:1 for the simplified model and 500:1 for the complex model respectively. These weights make the primary focus of the optimization to reach the target while preventing unrealistic accelerations and actuator inputs.

3.3.7 Additional Constraints

The system has a leg actuation length limit, which is included as an inequality constraint to limit the leg extension. In many cases this constraint causes the system to enter a flight phase, which poses an interesting challenge for convex optimizations and is discussed further in other sections.

3.4 Results

Simulations were run in MATLAB 2019b using the quadratic programming solver `quadprog` from the optimization toolbox. An MSI Trident 3 with an Intel i7-8700 processor, 16GB of RAM, and a Nvidia GeForce GTX 1060 was used for all the results presented. The constraints and cost function are generated before the control loop using the `MatlabFunction` code generator and symbolic toolbox. After each elapsed sample period, the states are sampled, the planning optimization is solved, and the planned control inputs are applied to the system dynamics–which are subsequently integrated with MATLAB's `ode45` medium-order differential equation solver.

3.4.1 Reliability Results

The objective is for all nodes along the trajectory to reach a desired height and horizontal velocity. Three studies are run to test the accuracy of forward hopping, vertical hopping, and the effect of increasing the model prediction time horizon. The first two tests utilize a time horizon of 1.5sec with a node-to-node δt and simulated sample time of 50ms which is greater than the loop computation time of $\sim$ 27ms.

Variable Desired Velocity. Forward velocities from 0.1m/s to 2.5m/s are added to the objective function at a constant hop height of 2m to test the controller's ability to match a single desired hop velocity across all nodes. Excluding the outliers (detailed in the discussion) the velocities generally correspond to the intended speeds.

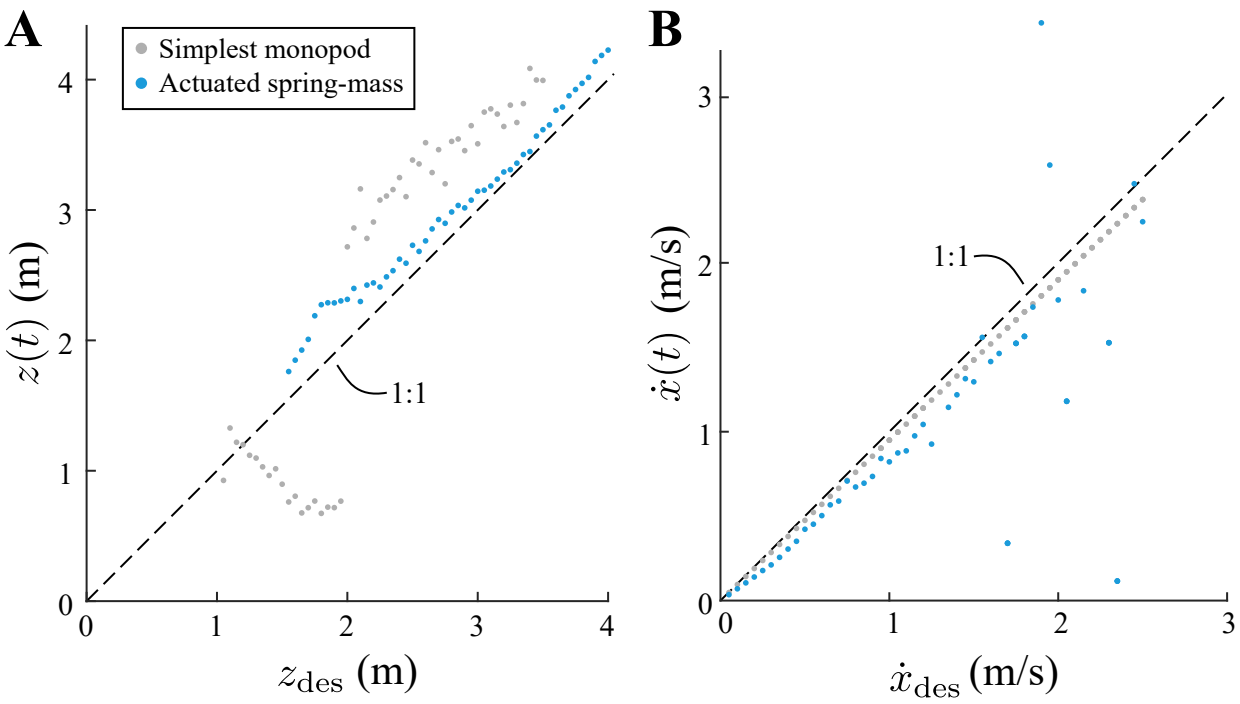

Figure 3.5: Reliability testing of both models w.r.t. their objective over 10 seconds of operating time **A.** The average gait cycle peak height **B.** The average gait horizontal velocity

Variable Desired Height. Similar to the last analysis, we try hopping in place. Both models use a desired height of 1.6 to 4.0 meters with no horizontal velocity. In many cases, we find that the optimal solution for the predictive horizon overshoots the desired state rather than reaching it exactly. Speculation of why this occurs is included in the discussion.

Variable Time Horizon. For the final analysis, we use a desired height of 2m and forward velocity of 0.5m/s with a variable time horizon. For consistency, this study keeps the δt constant at 50ms and varies the number of nodes as the horizon increases. The goal of this study is to investigate the gait behavior using a constant task but a changing horizon changes. We notice some variance, but in general, changing the time horizon does not have a large impact on the gait cycle.

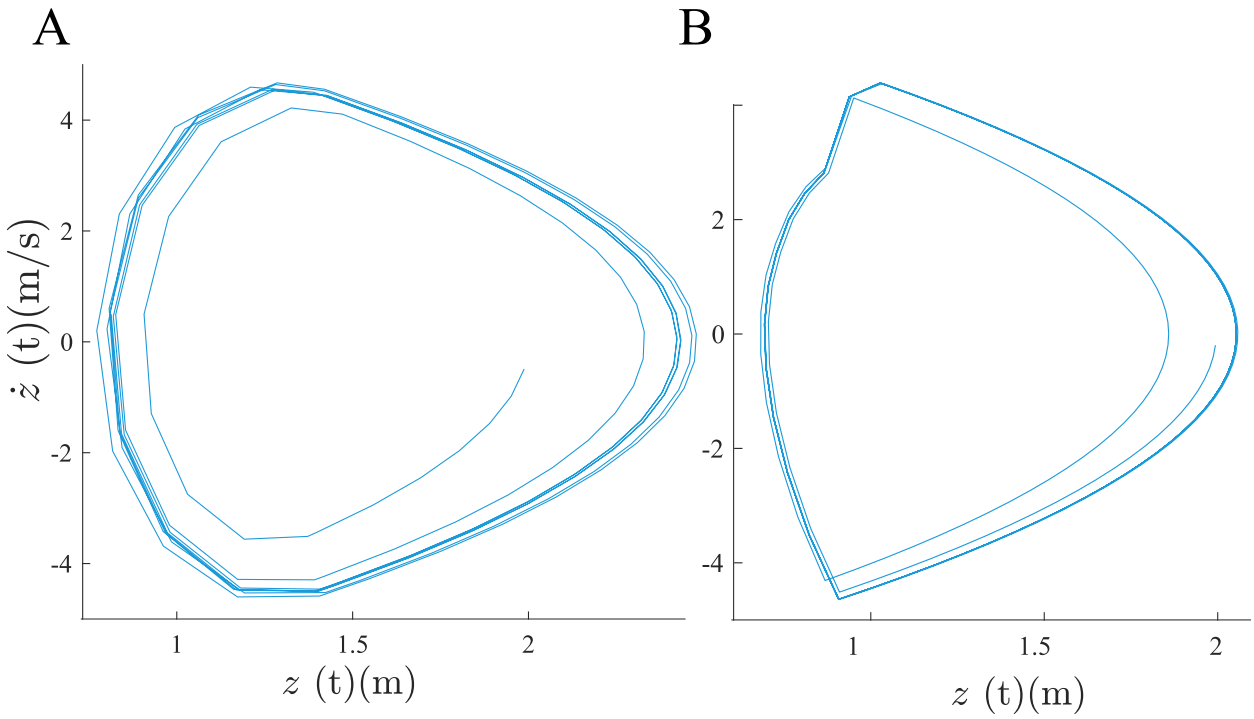

Figure 3.6: Two limit cycles of data points shown in Fig. 3.5 **A.** Simulation of the actuated spring-mass model **B.** Simulation of the simplest monopod

3.4.2 Transient Behaviors

Further investigating the gait behavior, we look at some of the phase plots developed with the aforementioned desired properties. We can see models start in flight with zero initial velocity and quickly converge to a stable limit cycle near the desired speed and mean hop height.

Disturbance Rejection. To test the abilities of the controller we subject the model to a 2m/s disturbance while in flight. The leg almost instantaneously adjusts, and the system recovers within a single stance phase to return to its steady state.

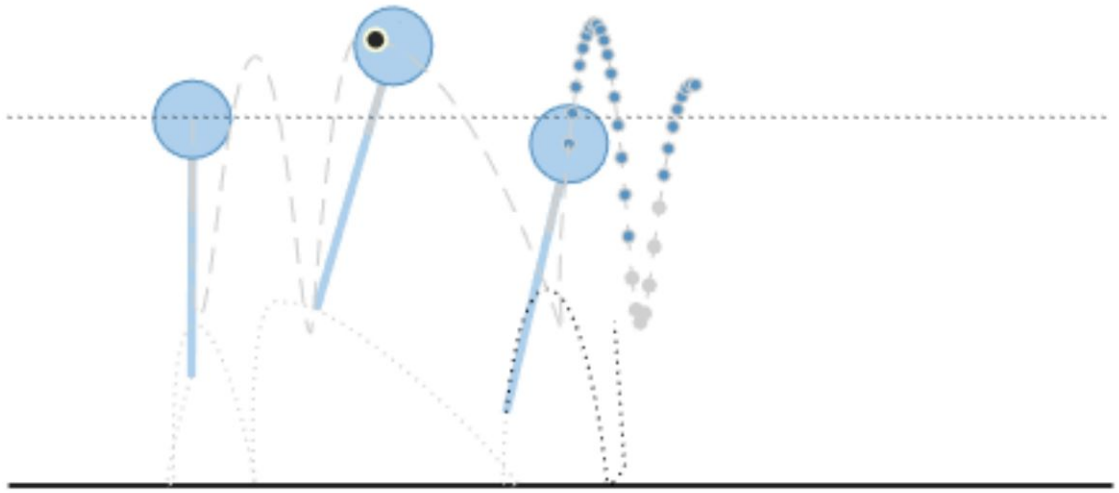

Figure 3.7: Simulated disturbance and reaction of the actuated spring-mass model to an instant velocity increase of 2m/s.

Change Directions. We further test the controller by changing the objective in the middle of a gait. While hopping forward, we reverse the desired velocity and introduce a new desired hop height. Immediately the generated gait changes, and the leg adjusts to match the new desired states. Once again the controller reacts within the first step to a stable cycle near the desired properties.

Stop in place. Finally, we modify the desired height mid-cycle to a length less than l_{max} with a horizontal velocity of zero. Almost immediately the ASLM model stops hopping and compensates to reduce the oscillation from the spring allowing it to remain at a stationary point.

3.5 Discussion

The nonlinear ASLM model formulation utilizes actuator deflections thereby generating spring forces which are used for control. Although optimizations typically minimize control input, this model's cost function is minimized by reducing the actuator acceleration. This leads the to trudging style gaits observed throughout the chapter. By keeping the actuator nearly stationary, the system can get both vertical and horizontal forces while in stance with minimal cost.

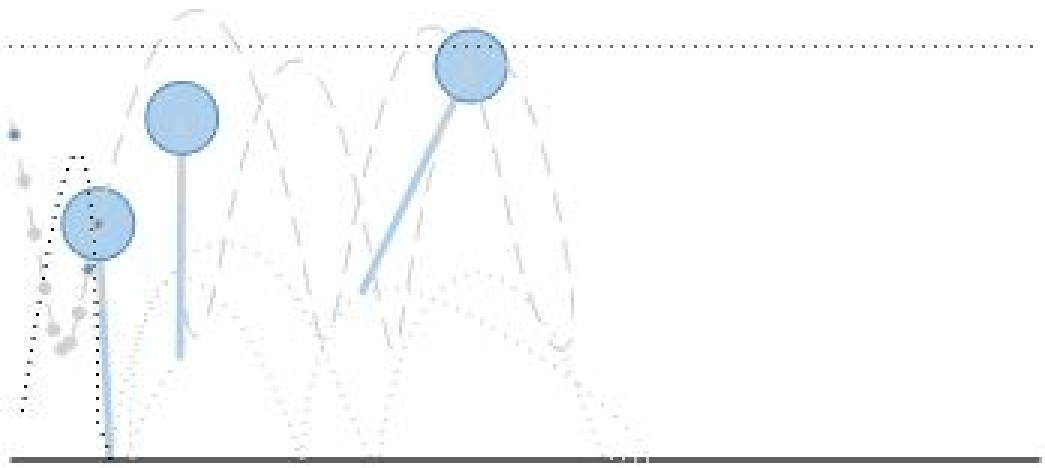

Figure 3.8: Systems reaction to a changed objective mid-gait to hop in the opposite direction at a faster pace and different height

.

Although we use 31 nodes throughout the reliability and reactivity tests, stable hopping has been maintained with as few as 5 nodes. As the nodes are reduced, the estimated trajectory becomes increasingly inaccurate leading to an obvious deviation from both the desired vs actual height (overshooting), and velocity (undershooting). The overshooting associated with vertical hopping is expected, as the optimization attempts to minimize the deviation between all nodes and the desired height, not the trajectory peak height. The velocity-based undershooting observed is likely a consequence of the multipart objective the system attempts to achieve (a forward velocity, vertical height, and minimal actuator motion).

3.5.1 Limitations

As is the case with many research topics, complications were present within specific instances in the simulations. For one, if the desired height used in the optimization is less than the max leg length, the inputs developed from the MPC will never force a flight phase. The system will instead "stretch" to the furthest point possible, and sway back and forth. This happens because the model never applies enough force to cause a flight phase and thus the model never utilizes the flight dynamics to achieve a more optimal solution. This behavior propagates to the multi-legged extension we formulated, where only one leg is utilized. For the single-leg model, all forces

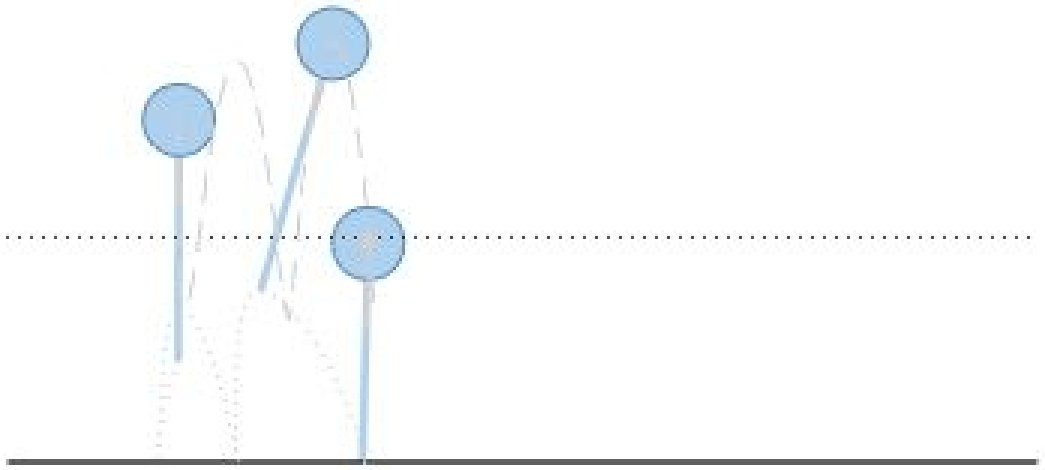

Figure 3.9: A hop-to-stand transition as a consequence of changing the desired height below l_{max}.

are generated from one source, because of this the optimizer realizes that this sole source will be accessible when the foot hits the ground. When we introduce a second leg, one leg is picked up, and never used again (*i.e.*, it hops around on one leg). Additionally, in some instances, the simulation becomes unstable. This happens a few times above a desired speed of 1.5m/s, and always at speeds > 2.5m/s (as seen in Fig.3.5B. Finally, instances of "procrastination" of the MPC are observed in some of the simplest model results. This is where a trajectory is planned utilizing both stance and flight, however, the control input extracted from the optimization never enforces this behavior. In term, the model instead either "slides" along the horizon or barely hops off the ground, as seen in Fig. 3.5A around the 1.5m desired height.

3.5.2 Future Work

Future work will extend and validate this method in 3D hopping. Further, we suspect that the method should be effective for multibody monopods with distal mass and body pitch. However, we anticipate that allowing torso rotations may require incorporating additional terms in the cost function to promote a level torso. Additionally, future work will explore through-slip conditions and assess the ability of the presented through-contact MPC to generate stable gaits both with and without a sliding foot.

3.6 Conclusion

We presented a convex motion planner for single-legged robots that generate behaviors without using predefined contact sequences or mode durations in real-time. Overall the results here show the method can control monopod locomotion without a predefined motion plan while keeping a problem convex. In planar simulations, we show that by only changing the desired states, stable limit cycles are developed as an emergent property of the controller. Additionally, the results indicate that limit cycles have limited dependency on the defined time horizon but are strongly shaped by the dynamics and task, suggesting that the MPC framework is flexible to the needs of dynamic monopods.

CHAPTER 4

MOTION PLANNING IN RESISTIVE MEDIA

Legged locomotion through viscous media is a less common topic of research. Furthermore, the existing methods for generating motion plans through these environments take minutes to compute, even for simple models. This is largely due to modeling the complex fluid interactions with the system. Our goal was to estimate these forces and generate motion plans on the fly. Using previous research we develop a similar model but find motion plans that minimize the cost of transport in seconds. The initial results are then tested on a hardware platform to demonstrate gait stability. This research is the first step towards a controller that will autonomously estimate the media properties to accurately model a robot's motion in viscous media. This section is a reprint of the conference paper named "Online Gait Optimization for Running in Resistive Media" [95] published in the 2022 International Conference on Robotics and Automation Engineering.
Specific contributions of this chapter include:

1. An online viable trajectory optimization (¿0.5Hz solve rate) for motion generation through a viscous media.
2. The average velocity at which the system's cost of transport is optimal.

4.1 Abstract

In-field legged robots need gaits and behaviors that allow them to run over and through all types of terrain, including through fluidic media such as ponds, bogs, and dunes. This chapter presents a method of rapidly generating legged running gaits, on-the-fly, for these resistive fluid-like terrains. To accelerate gait computation, we adapted a reduced-order dynamic model of legs in fluid fields for rapid optimization of energy-efficient motion plans. For the first time, we achieve online optimization for a legged model in resistive fluid-like media by formulating an optimal control problem with smooth and analytical cost functions and constraints. In numerical tests, we benchmark the computation times for our nonlinear optimization and validate that the optimized gaits follow a U-shaped speed versus cost curve commonly associated with energy-optimal legged locomotion. We further present a proof-of-concept demonstration of optimized gaits on a planarized monopod robot hopping through a bed of fluidized granular material – a surrogate fluid-like substrate.

4.2 Introduction

Animals can run and bound through flowing, granular, and deformable substrates such as sand, tall grass, water, and mud. Further, animals can adapt their gaits to fit the substrate in a manner that seemingly optimizes for speed or energy consumption. We want controllers for legged robotic systems that synthesize gaits through these terrains, which we lump together under the term "resistive media." This chapter extends state-of-the-art running control methods to include a computationally tractable fluid dynamic model to generate efficient locomotion through resistive media (Fig. 4.1A). We modify existing fluid dynamic models for legged locomotion to be both smooth and analytically differentiable, which allows us to generate running gaits in resistive media with a fast-solving optimization (< 2 seconds).

Running robot control dates back to tuned heuristics from the 1980's [72], but modern methods use model-based optimization to synthesize behaviors. While system models can be detailed with multibody dynamics [82, 37], reduced-order models can synthesize a variety of behaviors using simpler and faster computations [41, 24]. The Spring Loaded Inverted Pendulum (SLIP) model is a widely-used reduced-order model for running, reducing complexity by assuming a point-mass body and a massless linear spring leg [78]. The SLIP model is credited with inspiring successful leg control schemes [79, 28], but it is energetically passive and thus needs an extension to control behaviors with changing mechanical energy. Actuated variants of SLIP models can use continuous-time inputs to stabilize running behaviors across energy levels–even in rough terrain [5, 67].

Running in a resisting force field complicates the locomotion plant model and thus further challenges control. Fluids, granular materials (sand), foliage, and colloids (mud), are all governed by different physical phenomena that are principally modeled by expensive calculations (Navier-Stokes, discrete element methods, continuum mechanics, and multi-physics solvers, respectively). Approaches for tractable resistive modeling, such as the U-SLIP model [17] (or Underwater SLIP), took a lumped modeling approach for synthesizing SLIP model gaits in a resistive media generating underwater "punting" gaits [65]. While the U-SLIP model ignored resistive forces on the leg, a subsequent approach modeled leg drag and generated gaits through different fluid depths [3]. This model used a black-box particle swarm optimization to find gaits in > 10 minutes. More recently, this model was extended to include buoyancy and drag on the body and leg called the Fluid-Field SLIP (FF-SLIP) model, and used a Newton-Raphson search to identify gait parameters [7]. This method of trajectory generation accommodates a more complex dynamic model but requires some

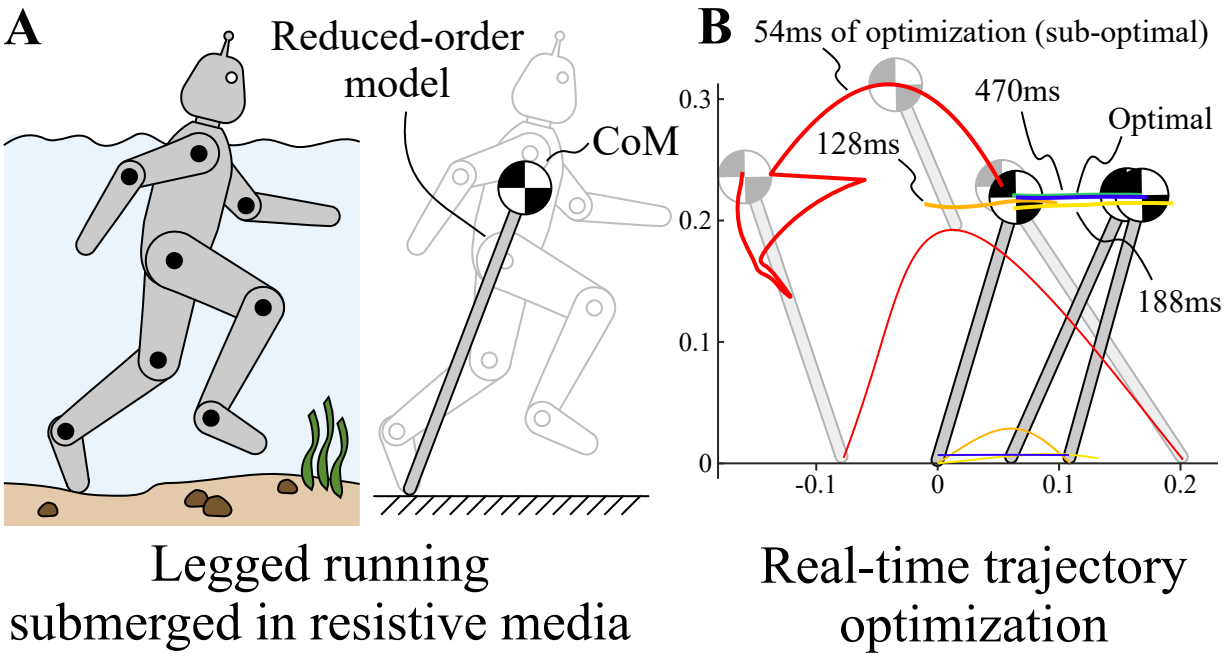

Figure 4.1: This chapter presents rapid optimization of legged locomotion in resistive media (*e.g.*, fluids) A. An illustration of how a multi-body legged robot is abstracted to a reduced-order FF-SLIP model. B. The Cartesian CoM and foot trajectories of a gait during optimization, achieving near-optimal behavior in 470ms.

gait parameters to be determined a priori (*i.e.*, duty cycle, phase velocity conditions, etc.) to converge and again results in solve times on the order of minutes.

This chapter details our approach toward generating running gaits online in resistive media in the following order. First, we detail the reduced-order fluid-field SLIP model [7] and present new modifications to its resistive media model to incorporate smooth analytical derivatives. Using direct transcription methods, we formulate this optimal control problem as a nonlinear program (NLP) and demonstrate that the smooth and easy-to-compute derivatives enable fast computation times ($<$ 2 seconds). When systematically commanding a range of desired gait speeds, the optimization finds a trade-off with energy cost that is smooth and U-shaped - a well-established feature of locomotion in robotics [98] suggesting that the optimization is successfully minimizing energy waste. As a final proof of concept, we implement an example optimized gait (Fig. 4.1B) on a monopod robot immersed in aerated granular media (a dry substitute for flowing liquids) to demonstrate stable hopping through resistive media with these fast-synthesized gaits (Fig. 4.5C).

4.3 Methods

Here we present a method of online gait optimization for SLIP models in resistive media–the FF-SLIP. Optimal gaits minimize the cost of transport based on motor effort by translating the reduced order SLIP torques τ_u and forces F_u to motor torques τ_M for a five-bar Minitaur leg. Smooth analytical derivatives allow us to quickly optimize gaits via large-scale direct collocation despite our nonlinear equations of motion.

4.3.1 Conventions

Sub-scripted capital letters refer to forces applied to either the body F_B or the leg F_L. Lowercase subscripts indicate the source of force F or torque τ, with the options being buoyancy b, gravity g, drag d, and control input u. A k subscript indicates an individual node in the optimization. Superscripts indicate flight q^F or stance q^S. The system has four degrees of freedom; leg angle θ, leg length l, and Cartesian body position x, y, with state vector q defined as

$$q = \begin{bmatrix} p & v \end{bmatrix}^T, \quad p = \begin{bmatrix} \theta & l & x & y \end{bmatrix}^T, \quad v = \dot{p}.$$

4.3.2 Dynamics

The presented method models both a single-legged robot and environmental forces using reduced-order dynamical approximations. Here, we derive a fluid-field forcing model that acts upon an

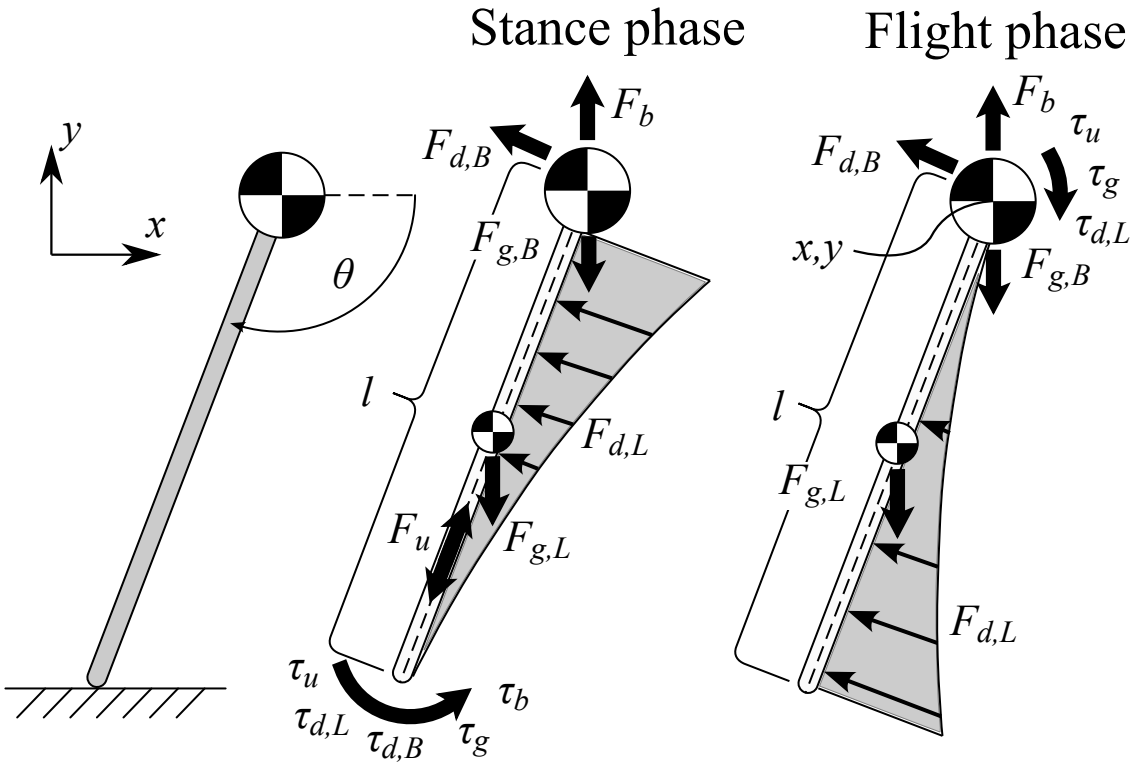

Figure 4.2: Diagram of the presented reduced-order legged model in stance (foot contact) and flight (no foot contact) phases.

actuated Spring-loaded Inverted Pendulum (SLIP) model. While the approximation makes several simplifying assumptions similar to [7], this reformulation novelly offers smooth analytical derivatives by assuming the system is fully submerged–thereby enabling fast online gait optimizations. Hopping/running requires changing contact modes, which we model independently as a stance (foot contact) phase and a flight (no foot contact) phase.

Stance phase. During stance, we assume the model has two degrees of freedom, leg rotation, and telescoping extension, and lump the body mass and leg mass together into a single rigid mass. The foot is pinned to the ground by a revolute joint and leg extension is modeled by a prismatic joint (Fig. 4.2). We assign polar coordinates to these degrees of freedom, a leg angle θ w.r.t. the horizontal plane and a telescoping leg length l. Control inputs during stance include a body torque τ_u and leg extension force F_u, which when combined with F_g and torque τ_g generated by gravity, we find:

$$\begin{aligned} F_g &= -\underbrace{m_B g}_{F_{g,B}} - \underbrace{m_L g}_{F_{g,L}} \\ \tau_g &= -gl(m_B + \frac{m_L}{2})\cos(\theta) \end{aligned} \tag{4.1}$$

where m_B and m_L are the body and leg masses respectively. We compute upward buoyant forces from the displaced weight of the fluid of density, ρ,

$$\begin{aligned} F_b &= \rho V_B g \\ \tau_b &= \rho V_B g l \cos(\theta). \end{aligned} \tag{4.2}$$

The FF-SLIP model drag on the system from the resistive media (the same method used here) results in a single force on the body, and two torques about the toe:

$$\begin{aligned} F_{d,B} &= -\frac{\rho C_B A_B \dot{l}|\dot{l}|}{2} \\ \tau_{d,L} &= -\frac{\rho C_L W_L \dot{\theta}|\dot{\theta}| l^4}{8} \\ \tau_{d,B} &= -\frac{\rho C_B A_B \dot{\theta}|\dot{\theta}| l^3}{2} \end{aligned} \tag{4.3}$$

where the body drag force, $F_{d,B}$, and drag torques, $\tau_{d,L}$ and $\tau_{d,B}$, are dependent on the system velocity. We note the leg actuation F_u does not experience drag because the drag force is always normal to the leg.

Incorporating these forces into the equations of motion with a torsional and linear actuation force results in the following equations of motion

$$
\begin{aligned}
(m_B + m_L)\ddot{l} &= F_u + F_{d,B} - F_g \sin\theta - F_b \sin\theta \\
(m_B + \frac{m_L}{3})l^2\ddot{\theta} &= \tau_u + \tau_{d,B} + \tau_{d,L} - \tau_g - \tau_b.
\end{aligned} \tag{4.4}
$$

Flight phase. While not in ground contact, we redefine our coordinates in a Cartesian frame with x and y position coordinates. Additionally, we no longer consider the forces acting along the leg actuation direction because the telescoping length does not produce a force on the body in flight. With our new reference frame, the gravitation force remains the same, but the torque becomes

$$
\tau_g = \frac{m_L g l \cos(\theta)}{2}. \tag{4.5}
$$

Similarly, the buoyancy force F_b remains unchanged and buoyancy torque τ_b is eliminated. Using a moving reference frame (the body Cartesian position) makes finding the drag forces and torques more challenging. The drag forces experienced by the system are based on the velocity of the system. In stance, we were able to assume the drag force was normal to the leg, however, we now have a moving reference frame and this assumption no longer holds. We instead find the velocity of the body CoM, v_L, as per:

$$
v_L = \dot{y}\cos\theta - \dot{x}\sin\theta, \tag{4.6}
$$

and use this to find the drag along the leg's length. Using this in conjunction with the body velocities $\dot{x}$ and $\dot{y}$ we find the directional drag forces and torques:

$$
\begin{aligned}
F_{d,B_x} &= -\frac{\rho C_B A_B \dot{x}|\dot{x}|}{2}, \quad F_{d,B_y} = -\frac{\rho C_B A_B \dot{y}|\dot{y}|}{2} \\
F_{d,L} &= -\frac{\rho C_L W_L l(\dot{\theta}|\dot{\theta}|l^2 + 3\dot{\theta}v_L l + 3v_L|v_L|)}{6} \\
\tau_{d,L} &= -\frac{\rho C_L W_L l^2(3\dot{\theta}|\dot{\theta}|l^2 + 8\dot{\theta}v_L l + 6v_L|v_L|)}{24}.
\end{aligned} \tag{4.7}
$$

Incorporating these forces with the torsional actuation force results in the following equations of motion:

$$
\begin{aligned}
(m_B + \frac{m_L}{3})l^2\ddot{\theta} &= \tau_u + \tau_{d,L} - \tau_g \\
(m_B + m_L)\ddot{x} &= F_{d,B_x} - F_{d,L}\sin\theta \\
(m_B + m_L)\ddot{y} &= F_{d,B_y} + F_{d,L}\cos\theta + F_g + F_b.
\end{aligned}
$$

Simulated Leg Parameters			
Parameter	Symbol	Value	Units
Body mass	m_B	1.36	kg
Leg mass	m_L	0.19	kg
Leg Thickness	W_L	0.0512	m
Body Volume	V_B	0.0002	m^3
Body Area	A_B	0.007	m^2
Environmental/Geometric Parameters			
Gravitational Acceleration	g	9.81	m/s^2
Media Density	ρ	997	kg/m^3
Body Drag Coefficient	C_B	0.47	-
Leg Drag Coefficient	C_L	1.15	-

Table 4.1: Fluid-field SLIP model (FF-SLIP) parameters

4.3.3 Assumptions

While the optimization methods can be broadly applied to running in resistive media generally, our presentation specifically uses kinematic and dynamic model parameters that match our testbed platform, the Minitaur leg (Fig. 4.5B). The model assumes buoyancy generates little to no force or torque on the system, and is therefore excluded. Contrary to the FF-SLIP model seen in [7], we assume the entire leg length is submerged. This assumption allows us to remove the surface swimming equilibrium factor and eliminates discontinuities in the dynamics associated with the leg being exposed to multiple substrates (*i.e.*, air and water) at the same time. We were also able to mathematically resolve the integral-based fluid drag force used for distributed dynamic loads over the variable leg depth. Leg acceleration $\ddot{l}$ is defined during flight via a forced second-order model using an actuation force F_u and an added damping term c per:

$$m_L\ddot{l} = F_u - c\dot{l} - m_L g. \tag{4.8}$$

Drag forces heavily depend on the leg length, so allowing actuation during flight lets the optimization determine an optimal leg length to minimize the cost of transport. The damping term discourages unrealistic instantaneous changes in leg length. To smooth the absolute value terms for analytical differentiation, we use the approximation $\|z\| \approx \sqrt{z^2 + \epsilon^2}$ where $\epsilon \ll 1$.

4.3.4 Optimal Control Problem

The optimal control problem is cast as a nonlinear direct collocation trajectory optimization [85].

$$\begin{aligned} \min_{q,u,t} \quad & f(q,u,t) \\ \text{s.t.} \quad & c_1(q,u,t) = 0 \\ & c_2(q,u,t) \leq 0 \end{aligned} \tag{4.9}$$

where q is the system state vector $q_k = [p_k\ v_k]^T$ with $p_k = [\theta_k\ l_k\ x_k\ y_k]^T$, u is the control input vector $u_k = [F_{u,k}\ \tau_{u,k}]^T$, and t is the total phase duration. Each state and input is represented as a vector of N decision variables for $k \in \{1:N\}$. The equality $c_1(q,u,t)$ and inequality $c_2(q,u,t)$ constraints (detailed below) encode the system dynamics, continuity between modes, and task requirements. We then attempt to minimize the overall cost of transport in the cost function $f(q,u,t)$ across both modes (stance and flight).

Dynamics Constraints. To enforce the system dynamics on the evolution of state variables, we add node-to-node defect constraints using Explicit Euler integration ($N = 101$ nodes per mode) with state derivatives per the previous section.

$$q_{k+1} = q_k + \dot{q}_k \Delta t, \quad \Delta t = \frac{t}{N-1}$$

The two separate modes are connected using continuity constraints.

Continuity Constraints. SLIP models inherently have discontinuous dynamics when switching from stance to flight. To address this, we create a mode schedule for the trajectory optimization, with the first mode being ground contact and the second having no ground contact. To enforce continuity at the interchange point, we use equality constraints to make the final mode 1 states equal to the initial mode 2 states. Enforcing gait periodicity is achieved similarly, by constraining the initial mode 1 states to equal the final mode 2 states,

$$\begin{aligned} 0 &= q^S_{final} - q^F_{init} \quad \text{(Mode continuity)} \\ 0 &= q^F_{final} - q^S_{init} \quad \text{(Periodicity)}. \end{aligned}$$

Kinematic Constraints. Additional constraints are added to limit the operating range of our system, and match kinematic limits using the 0.1m upper bar lengths l_1 and 0.2m lower bar lengths l_2 of the Minitaur testbed platform,

$$\begin{aligned} 0.1 \leq l \leq 0.3 \quad & \text{(Leg length limit)} \\ -\frac{3\pi}{4} \leq \theta \leq -\frac{\pi}{4} \quad & \text{(Leg angle limit)} \\ 0 \leq y + l\sin(\theta) \quad & \text{(Toe above ground)}. \end{aligned}$$

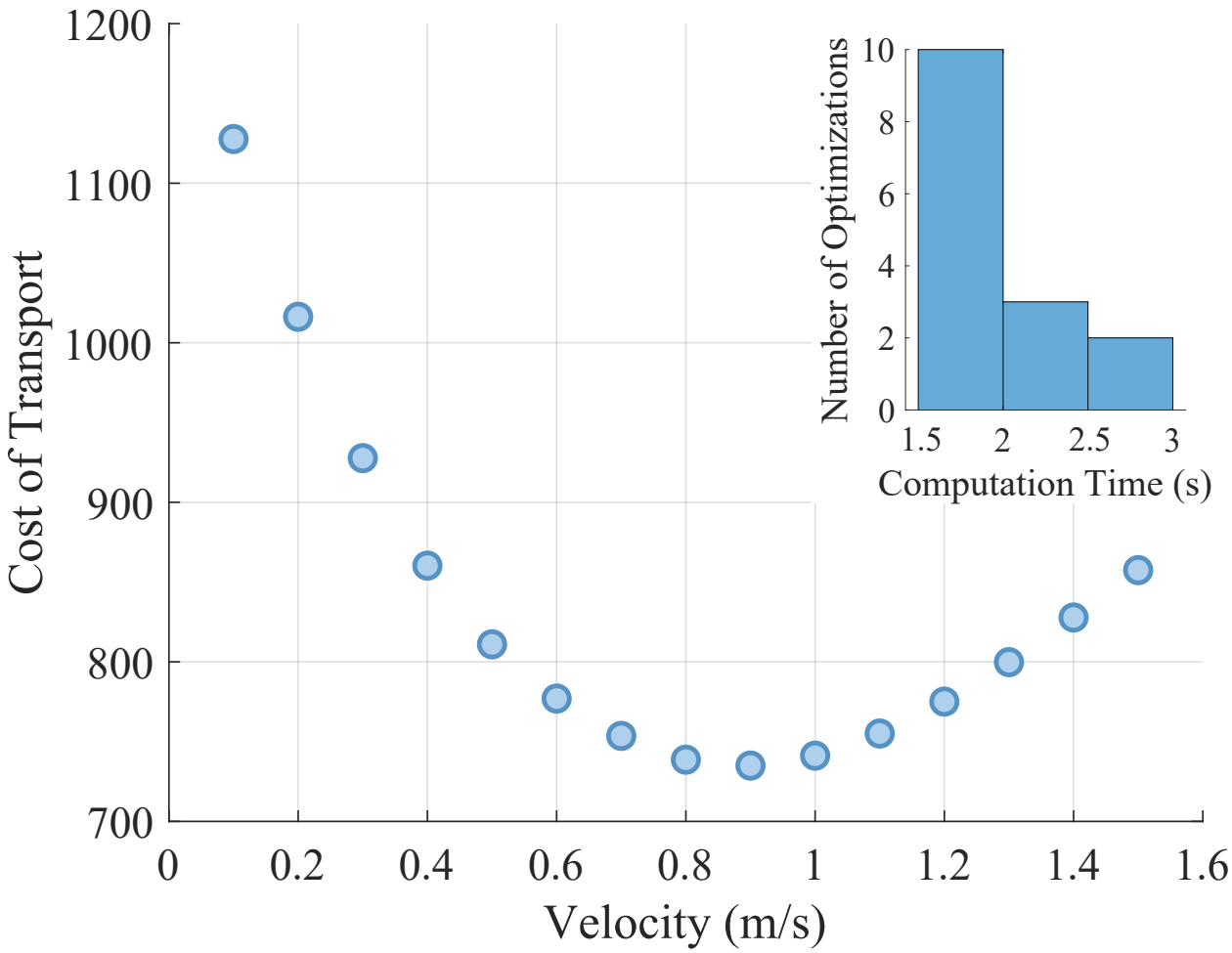

Figure 4.3: Cost of transport and solve times for gaits of varied speeds. The COT curve is smooth and U-shaped – a common feature of energy-minimizing locomotion that suggests the optimization is avoiding pseudo minima.

Task Constraints. We add task constraints to prevent trivial zero-duration gaits (duration $> 0.001s$), and further prevent backward locomotion and allow only unilateral contact forces:

$$0.1 \leq x_{final} - x_{init} \qquad \text{(Forward motion)}$$

$$0 \leq F_l \qquad \text{(Unilateral leg force).}$$

Control Input Constraints. Limiting the control inputs to remain within our hardware limits is critical to successfully transitioning model-based control to a hardware platform. Motor torques are calculated via constraints that map the reduced-order FF-SLIP (Fig. 4.2) to a five-bar leg morphology (Fig. 4.4A) using a kinematic relationship. We use the law of cosines to find interior leg angles

$$\cos(\phi) = \frac{l_1^2 + l_2^2 - l^2}{2l_1 l_2}, \quad \cos(\psi) = \frac{l_2^2 + l^2 - l_1^2}{2l_2 l} \tag{4.10}$$

where ϕ is the angle between the virtual leg and the upper link, and ψ is the angle between the upper and lower leg links. Solving for ϕ requires a $\cos^{-1}$ term which frequently causes optimization infeasibility, so we introduce slack variables $s_1 = \cos(\phi)$ and $s_2 = \cos(\psi)$. Using equality constraints $s_1 - \cos(\phi) = 0$ and $s_2 - \cos(\psi) = 0$ allows us to reliably find optimal solutions. We then estimate the torques using

$$\tau_{M1} = \frac{F_u l_1}{2\cos\phi\sin\psi} + \frac{\tau_u}{2}, \quad \tau_{M2} = \frac{F_u l_1}{2\cos\phi\sin\psi} - \frac{\tau_u}{2} \tag{4.11}$$

and limit these torques based on the T-Motor U10 KV100 maximum torque per:

$$|\tau_{M1}|,\ |\tau_{M2}| \leq 5.91. \tag{4.12}$$

Compared to torques computed with the Jacobian relation, $[\tau_{M1}\ \tau_{M1}]^T = J(l)[F_u\ \tau_u]^T$, the estimates are conservative.

Cost Function. As previously mentioned, we optimize for cost of transport (CoT) per:

$$\text{CoT} = \left(\int_0^t |\tau_{M1}| + |\tau_{M2}| dt\right) / (x_{final} - x_{init}). \tag{4.13}$$

We then export the NLP's costs, constraints, and their analytical derivatives using the MATLAB framework COALESCE [44]. We then solve the optimization using the open-source large-scale NLP solver, IPOPT [13].

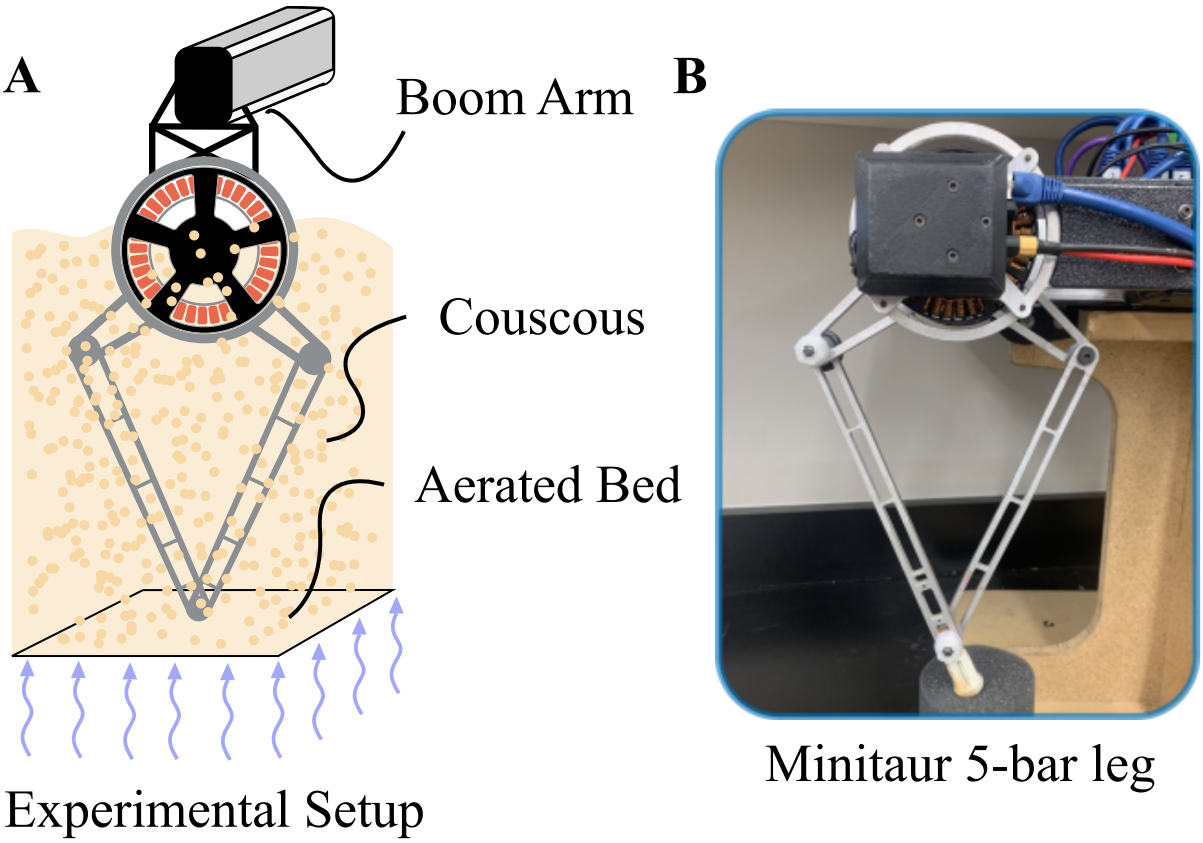

Figure 4.4: A. Schematic of a two-motor five-bar monopod robot planarized by rigid attachment to a 2-DOF boom arm. Fluid resistance is emulated by an aerated bed of dry granular media (couscous). B. A five-bar "Minitaur" robot leg.

4.4 Results

4.4.1 Numerical Results

Using the parameters seen in Table 4.1, we generate trajectories that minimize the energetic cost of transport. We specify a target speed for each optimization by constraining the final velocity during flight. By generating optimal trajectories with increasing desired velocities by 0.1 m/s increments, we see the model's optimal running velocity occurs nearest to 0.9 m/s. Re-running the optimization without a prescribed velocity confirms the lowest-cost speed to be 0.887 m/s. The histogram included in this plot shows the majority of optimizations found the solution in under 2 seconds, demonstrating online computation speeds. One consistent feature of the optimized gaits is that almost no hip torque is required, which contrasts with previous FF-SLIP research finding actuating τ_u during swimming is optimal, producing a kick-back motion.

4.4.2 Hardware Results

We further test the model-based results by implementing the optimal CoT gait in a hardware hopping experiment. The monopod is a 2 DOF constrained Minitaur leg in an aerated couscous bed, as illustrated in Fig. 4.5A, similar to [59]. This setup simulates a viscous fluid media without exposing the leg to normal risks associated with submerging electrical components in liquid. Our initial findings (after only a handful of tests) show additional constraints are needed to generate stable gaits. Although the optimal CoT gaits were able to move the leg through the media, the foot regularly slipped when in contact with the aerated bed. Additional constraints to enforce longer stance times and no instantaneous torques at touchdown/takeoff led to stable periodic motion of the leg resembling the gaits in simulation, as depicted in tiles in Fig. 4.5.

4.5 Conclusion

We presented a framework for optimizing SLIP model trajectories through resistive media during operation ($\sim 1Hz$). Furthermore, we generated trajectories at different desired velocities and tested the optimal cost of transport trajectory on a hardware platform. The results suggest that, with the presented parameters, there is a globally optimal gait minimizing cost of transport that can be found in under two seconds. Additionally, we show that the presented energy-efficient gaits are also stable when preliminarily tested on hardware with a robotic "Minitaur" leg.

Ongoing work will extend the formulation to include running with transitions between media (*e.g.*, half in air and half in water). Using differentiable functions to represent fluid-air boundaries

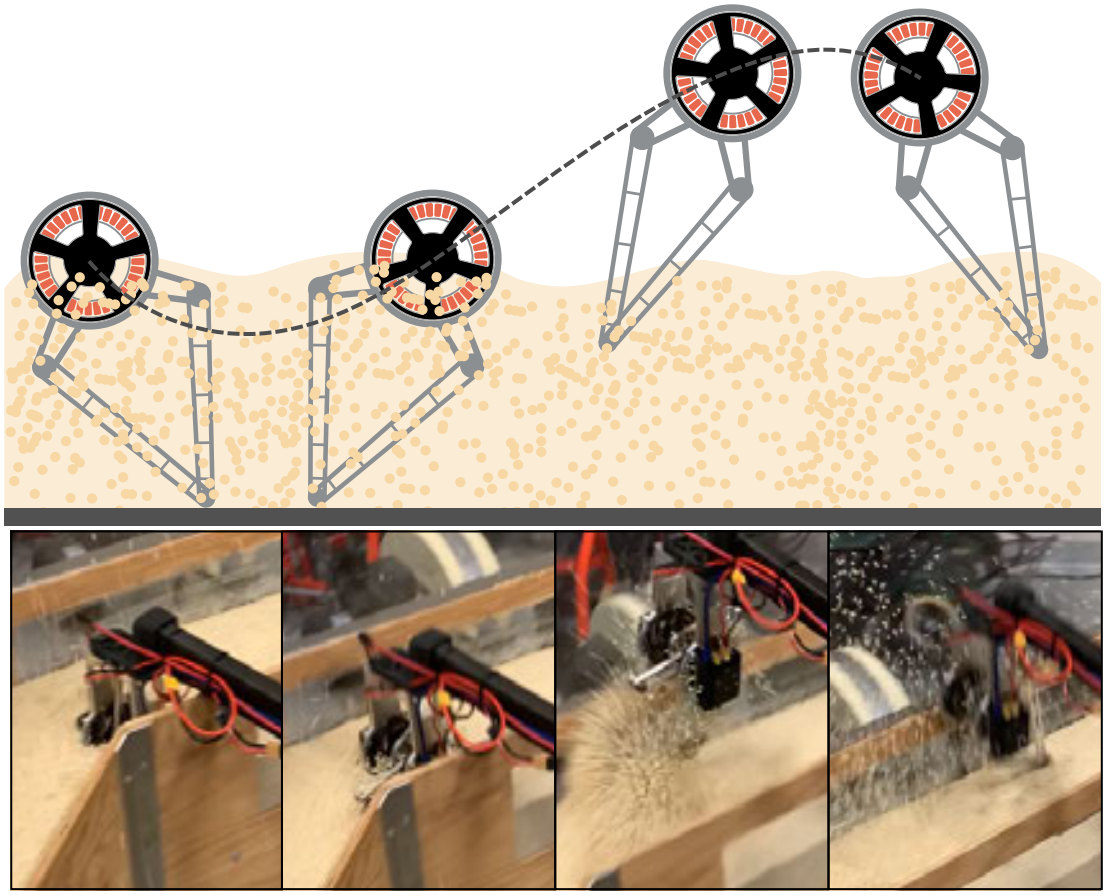

Figure 4.5: Demonstration of fast-optimized monopod hopping through aerated couscous– a surrogate fluid-like substrate.

(*e.g.*, sigmoids) will preserve the online optimization speeds while making the resistive drag forces from the media more precise. Measuring deviations between simulation and hardware will both 1) provide insight into the accuracy of our model and 2) allow the model to modify the drag coefficients via online regression. Taking advantage of the solve time, we aim to generate gaits on-the-fly to match these online-fitted models, enabling rapidly adaptive behaviors to new media *in situ*.

CHAPTER 5

DYNAMIC OBSTACLE AVOIDANCE

Obstacle avoidance is necessary for autonomous platform navigation. In most instances obstacles are static, allowing the planner to map and avoid its surroundings. When the obstacles are moving along unknown paths, avoidance and planning become much more challenging. We devised a solution which works with multiple systems and rapidly re-plans around obstacles to a goal in milliseconds of computation time. By estimating the dynamics of our system and penalizing close obstacle proximity, we find optimal motion plans using a model predictive controller (MPC). This section is a reprint of the conference paper named "Avoiding Dynamic Obstacles with Real-time Motion Planning using Quadratic Programming for Varied Locomotion Modes" [94] published in the 2022 International Conference on Intelligent Robots and Systems.
Specific contributions of this chapter include:

1. A soft piecewise constraint that penalizes the cost function when an agent passes an obstacle's "at risk" distance–as computed by the half-space relaxation
2. A numerical and physical parameter study that quantifies the reliability of the soft constraint formulation
3. A validation of the method on multiple simulated systems with varying dynamics and quadrotor hardware in dynamic avoidance tasks.

Abstract

We present a real-time motion planner that avoids multiple moving obstacles without knowing their dynamics or intentions. This method uses convex optimization to generate trajectories for linear plant models over a planning horizon (*i.e.*, model-predictive control). While convex optimizations allow for fast planning, obstacle avoidance can be challenging to incorporate because Euclidean distance calculations tend to break convexity. By using a half-space convex relaxation, our planner reasons about an approximated distance-to-obstacle measure that is linear in its decision variables and preserves convexity. Further, by iteratively updating the relaxation over the planning horizon, the half-space approximation is improved, enabling nimble avoidance maneuvers. We further

augment avoidance performance with a soft penalty slack-variable formulation that introduces a piecewise quadratic cost. As a proof of concept, we demonstrate the planner on double-integrator models in both single-agent and multi-agent tasks–avoiding multiple obstacles and other agents in 2D and 3D environments. We show extensions to legged locomotion by bipedally walking around obstacles in simulation using the Linear Inverted Pendulum Model (LIPM). We then present two sets of hardware experiments showing real-time obstacle avoidance with quadcopter drones: (1) avoiding a 10m/s swinging pendulum and (2) dodging a chasing drone.

5.1 Introduction

Planning with avoidance is necessary for robots in real-world environments. Whether in factories, hospitals, or exploring the outdoors, robots need to avoid collisions with obstacles or other agents without advance knowledge of their intentions. Further, as robots become increasingly fast and their environments more dynamic, motion planning must be expedited to keep pace. This work presents a convex motion planning optimization capable of avoiding dynamic obstacles without knowing their intentions that solves in real time (on the order of 1kHz). The computation speed and reliable optimization convergence make the planner suitable for highly dynamic, real-world applications.

A seminal method for avoiding dynamic obstacles is to detect and avoid a collision cone as computed by the motion of "velocity obstacles" [30]. Optimal Reciprocal Collision Avoidance (ORCA) builds atop this concept to find collision-free motions. It does so by finding sets of velocities for each agent that would create collisions and disallow them. These methods scale well to many agents and do not require explicit communication, but instead make assumptions about other agents to guarantee collision-free strategies [12] which have since been unified across agent plant dynamics [11]. However, these guarantees are less reliable under the practical actuation limitations of robots. Other methods explicitly use mutual communication to update collision-free trajectories [9]. Further, safety-focused control methods like barrier functions [4] use linear constraints to divert agents away from failure modes such as obstacles without explicitly treating them as agents.

Optimization approaches aim to use real-time motion planning (*i.e.*, Model Predictive Control, or MPC) to flexibly generate control on-the-fly for quadrotors [10], fixed-wing aircraft [83], quadrupeds [24], bipeds [70]. Further, these methods can be more robust to dynamic assumptions

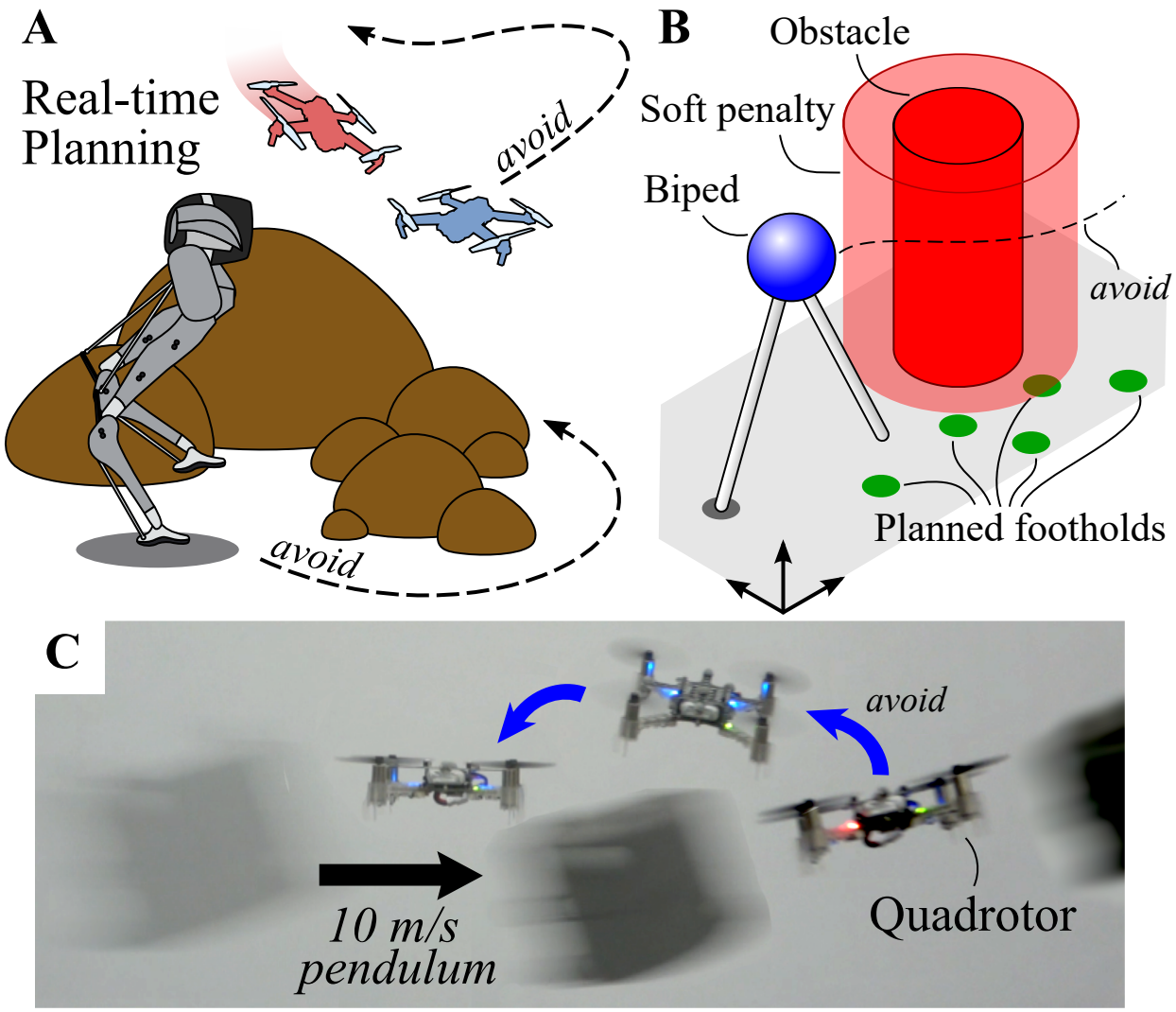

Figure 5.1: **A.** Robots must quickly plan their motions to avoid obstacles and adversaries in real-world environments. **B.** Our optimization methods use linear constraints and costs to achieve reliable real-time motion planning that spans locomotion modes from flying to walking. **C.** Demonstration of a quadrotor evading a 10m/s swinging pendulum. Note: drone positions were adjusted for image clarity, see supplementary video for unadjusted footage.

about obstacles. MPC has the additional benefit of minimizing a cost function while respecting constraints, (*e.g.*, physical limits, and obstacles [21]). To be effective, the motion planners within MPC must be sufficiently fast and reliable to react in real-time, and thus turn to fast and reliable convex optimizations [16]. While MPC typically assumes convex linear constraints, the Euclidean distance constraints inherent to obstacle avoidance are nonlinear and thus impede convexity. Convex relaxations have formulated convex SOCPs [86] which solve at 8-30Hz, and fast nonlinear solvers like SQP [48] and embedded solvers like PANOC [76] and have been practically successful in achieving kHz control rates.

We seek an MPC formulation that leverages well-developed fast convex solvers, specifically for quadratic programs (QPs), that reliably avoid dynamic obstacles and adversaries. QP solvers are well-developed and known to have kHz computation rates [84]. We leverage a half-space convex relaxation of the obstacle-free space [20] and iteratively update that approximation over the MPC planning horizon. Using a slack-variable formulation, we add a penalty term that serves as a fast-solving piecewise quadratic cost. In this chapter we show that this real-time approach to dynamic obstacle avoidance can nimbly avoid dynamic obstacles and adversaries in a manner that:

1. Scales to multiple obstacles and 3D
2. Handles non-cooperative agents and adversaries
3. Extends to varied locomotion modes (*e.g.*, bipeds)
4. Is effective on hardware in real-time.

5.2 Methods

Here we present a method of dynamic obstacle avoidance using a convex MPC motion planning framework. We show that a half-space representation of obstacles is sufficient for practical real-time avoidance, given the avoidance is treated as a penalty rather than a hard constraint. Using the previous motion plan, piecewise constrained slack variables penalize the agent when it passes an obstacle distance threshold.

5.2.1 Conventions

The following notation is used throughout the paper: A right subscript of k refers to node k along the trajectory where $k \in 1, 2, ..., N$ and N is the number of nodes. $R, R^n, R^{n \times m}$ represent

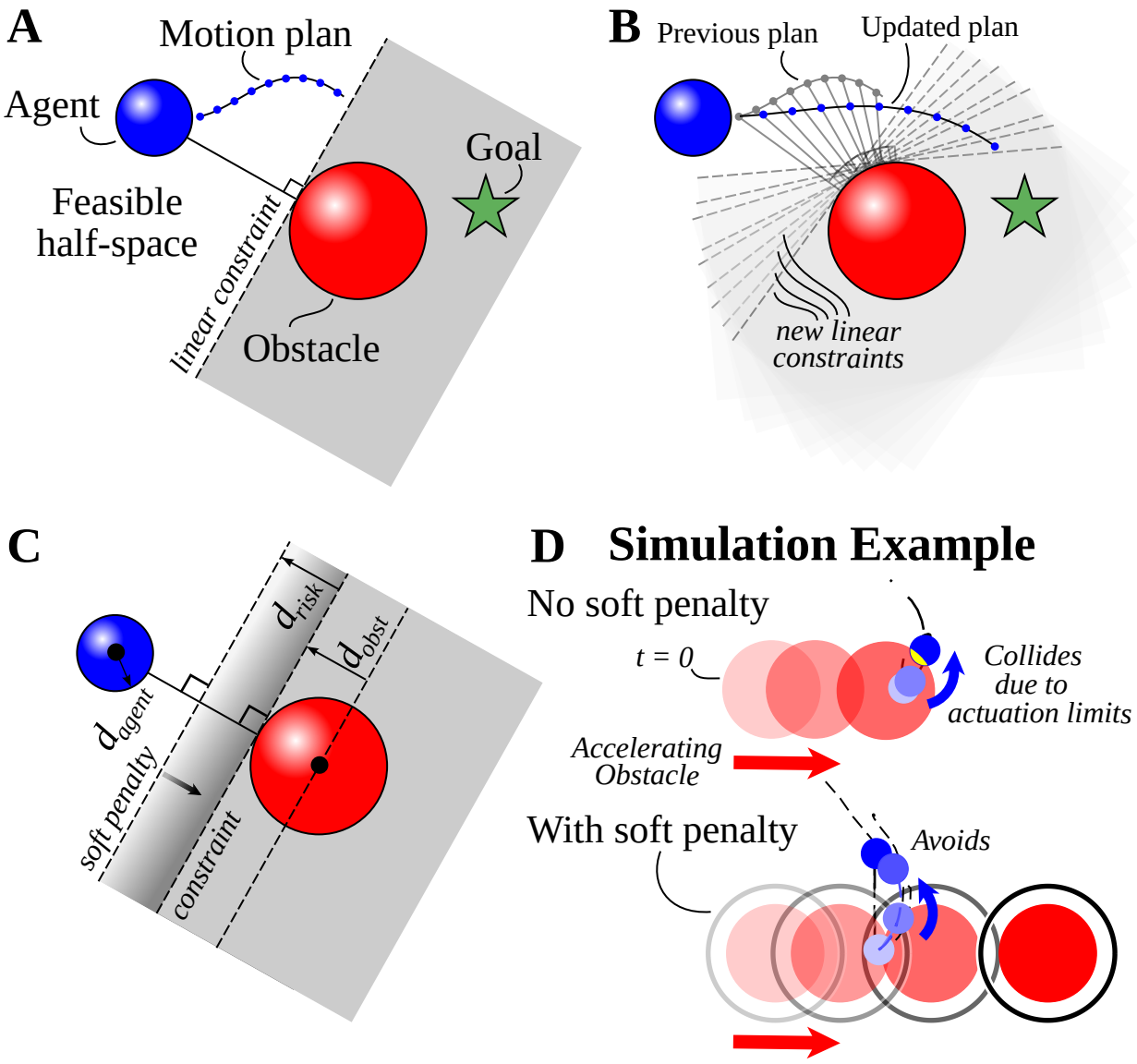

Figure 5.2: **A.** A half-space relaxation slices the feasible space in half with the plane tangent to the closest obstacle point. **B.** Half-spaces are recut using the previous planned trajectory when computing the closest obstacle point, iteratively improving the half-space approximation. **C.** We add a quadratic cost once the agent crosses a threshold close to the half-space cut. **D.** In simulation, this piecewise cost improves avoidance when the agent has physical limits.

the space of real numbers, vectors with length n, and matrices with n rows and m columns. Scalar values are lowercase and italicized letters (*e.g.*, $x \in R$). Bold, lowercase letters represent vectors (*e.g.*, $\mathbf{p}_k \in R^n$) while bold, uppercase letters represent matrices (*e.g.*, $\mathbf{P} \in R^{n \times m}$). Variables $\mathbf{p}_k$, $\mathbf{v}_k$, $\mathbf{u}_k$, δ_k are positions, velocities, control inputs, and slack variables with the robot state vector defined as $\mathbf{q}_k = [\mathbf{p}_k \ \mathbf{v}_k]^T$ and its derivative $\dot{\mathbf{q}}_k = [\dot{\mathbf{p}}_k \ \dot{\mathbf{v}}_k]^T$. The double integrator model is represented in three dimensions R^3, and the LIPM in two dimensions R^2. Right subscripts with a comma $\mathbf{q}_{prev,k+1}$ indicate a description of the variable, followed by the node number. A star superscript indicates state variable k along the trajectory with $\mathbf{p}^*_{\text{agent},k}$ being the agent's most recent planned trajectory and $\mathbf{p}^*_{\text{obst},k}$ being the estimated obstacle trajectory. $\mathbf{W}_\star$ denote positive semi-definite symmetric weighting matrices for cost category, $\star$.

5.2.2 Assumptions

We assume the initial positions $\mathbf{p}_1$ and velocities $\mathbf{v}_1$ of both the agent $\mathbf{q}_{\text{agent}}$ and obstacle $\mathbf{q}_{\text{obst}}$ are known. As is common in most MPC implementations, we assume linear plant dynamics. Although the obstacles are simulated using double integrator dynamics, our avoidance constraint formulation Sec. 5.2.4 uses only the obstacle's initial states $\mathbf{q}_{\text{obst},1}$ with a simple single integrator assumption of the dynamics. Specifically, the obstacle's future motion is estimated using only its current states $\mathbf{q}_{\text{obst},1}$ integrated forward in time, ΔT, per:

$$\mathbf{p}^*_{\text{obst},k+1} = \mathbf{p}_{\text{obst},1} + \mathbf{v}_{\text{obst},1} \ dt \ \mathbf{k}, \quad \mathbf{k} = \{1, 2, ..., N-1\} \tag{5.1}$$

Despite these simplifying model assumptions, we will demonstrate its effective avoidance behaviors even when implemented in crowded multiagent scenarios (Sec. 5.3.4). In the presented multi-agent and multi-adversary scenarios, there is no communication of plans or intent across agents–and adversaries' actual motion is generated using second-order system dynamics with a PD control law to model chasing behavior.

5.2.3 Model Predictive Control

MPC is a common method of using online optimizations to track desired trajectories but can also be used for online MPC is a common method of using online optimizations to track desired trajectories but can also be used for online motion generation with loose (or non-existent) tracking requirements. We use it to generate the robot state vectors $\mathbf{q}_k$ and associated control inputs $\mathbf{u}_k$ while avoiding obstacles–facilitated in part by including a slack variable δ_k.

As is typical in MPC, we facilitate model prediction in our equality constraints using an Explicit Euler integration of our system dynamics. At each control loop, we constrain the measured states to equal the initial trajectory states. The optimization uses a multi-part quadratic cost, and linear constraints per:

$$\begin{aligned}
\min_{\mathbf{q}_k,\mathbf{u}_k,\delta_k} \quad & \underbrace{J(\mathbf{q},\mathbf{u},\mathbf{s})}_{\text{Task}} + \sum_{k=1}^{N} \underbrace{||\delta_k||^2_{\mathbf{W}_{\text{avoid}}}}_{\text{Avoidance}} \\
\text{s.t.} \quad & \dot{\mathbf{q}}_k = \mathbf{A}\mathbf{q}_k + \mathbf{B}\mathbf{u}_k && \text{(Dynamics)} \\
& \mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k dt && \text{(Model Prediction)} \\
& f(\mathbf{q}_k, \delta_k) \leq 0 && \text{(Avoidance)} \\
& h(\mathbf{q}_k, \mathbf{u}_k) \leq 0 && \text{(Task Constraints)}
\end{aligned}$$

and is detailed in later sections, where the cost function $J(\mathbf{q},\mathbf{u},\mathbf{s})$ has a corresponding weighting matrix, $\mathbf{W}$, and is specific to each task and model (J for brevity).

5.2.4 Convex Obstacle Avoidance

Determining an avoidance strategy for even a simple circular object is inherently a nonlinear problem by the nature of the Euclidean distance formula. To avoid this nonlinearity, we instead relax the Euclidean distance using a half-space representation (or relaxation) of the distance constraint. We draw a plane that bisects the space into feasible and infeasible regions by finding the obstacle point nearest to the agent and defining a cutting plane tangent to it Fig. 5.2A. Distance to this plane is a linear function of our design variables and thus can be formulated as convex linear constraints.

These half-space relaxations, while convex, have the potential to overly limit the mobility of the agent. However, we can improve this approximation by iteratively updating the planar cuts along the planned trajectory by using the previously generated motion plans as seen in Fig. 5.2B. To accomplish this we find the distance between a point of interest on the agent and an obstacle via:

$$\mathbf{d}_k = \mathbf{p}^*_{\text{agent},k} - \mathbf{p}^*_{\text{obst},k} \tag{5.2}$$

Since the vector is from the agent and obstacle point of interest, we need to scale this vector to find the distance to a collision. For simplicity this paper only considers the agent and obstacles to be spherical, however, this obstacle can take on different shapes. The constraint then becomes:

$$\mathbf{d}_k \cdot (\tilde{\mathbf{d}}_k - \mathbf{p}_k) \leq 0. \tag{5.3}$$

where $\tilde{\mathbf{d}}_k$ is the offset necessary to represent the obstacle and agent width along $\mathbf{d}_k$. This is calculated as $\tilde{\mathbf{d}}_k = (d_{\text{agent}} + d_{\text{obst}})\hat{\mathbf{d}}_k$ where d_{agent} and d_{obst} are simply the respective radii of the obstacle and agent for spherical objects, and $\hat{\mathbf{d}}_k$ is the unit vector of $\mathbf{d}_k$.

5.2.5 Soft Penalty

This method alone only produces a rough estimate of the obstacle future states, and thus commonly violates constraints leading to infeasible optimizations. To resolve this we define an additional "at risk of collision" distance d_{risk} which acts as a buffer to penalize the optimization when the agent enters this space. We develop a second planar cut, parallel to the first, but include an additional design variable in the equation Fig. 5.2C. This design variable δ_k is a slack variable, and will penalize the optimization based on the distance between the obstacle and agent. We bound this penalization by including a second constraint on the slack variable. This piecewise formulation ensures there is no penalty outside of the soft penalty line, but escalates quickly after crossing the soft penalty threshold. These avoidance constraints for $f(\mathbf{q}_k, \delta_k)$ are written as such:

$$\begin{aligned} -\delta_k &\leq 0 \quad \text{(No-penalty Region)} \\ \mathbf{d}_k \cdot (\tilde{\mathbf{d}}_k - \mathbf{p}_k) - \delta_k &\leq 0 \qquad \text{(Penalty Region)} \end{aligned} \tag{5.4}$$

where $\tilde{\mathbf{d}}_k = (d_{\text{agent}} + d_{\text{obst}} + d_{\text{risk}})\hat{\mathbf{d}}_k$, and the slack variable δ_k is included in the cost function to enforce the penalties associated with the agent encroaching on the obstacle's defined threshold,

$$J_{\text{avoid}}(\delta) = \sum_{k=1}^{N} ||\delta_k||^2_{\mathbf{W}_{\text{avoid}}}. \tag{5.5}$$

This general avoidance formulation scales well computationally when expanded to many obstacles so long as the plant dynamics are approximated as linear–examples of which are detailed in the sections that follow.

5.2.6 Double Integrator Model Example

We introduce the state space model for a double integrator (Eq. 5.6) as a minimalist representation of mechanical locomotor dynamics. We use it to both simulate the avoidance method and to generate trajectories online for our quadrotor hardware. The dynamic model we use allows each Cartesian direction to be controlled, resulting in the state space model:

$$\dot{\mathbf{q}}_k = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{q}_k + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{u}_k. \tag{5.6}$$

In addition to the constraints presented in the previous section, the model has both cartesian bounds $h(\mathbf{q}_k)_1 = \mathbf{p}_{min} \leq \mathbf{p}_k \leq \mathbf{p}_{max}$ and control input bounds $h(\mathbf{u}_k)_2 = \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max}$. The cost function for this model and the LIPM (presented in the next section), have significant overlap and thus are detailed in Sec. 5.2.8.

5.2.7 Linear Inverted Pendulum Example

The Linear Inverted Pendulum Model (LIPM) has been widely used in control design for legged robots, particularly motion generation [46]. It has been effective in capturing the essential dynamics of bipedal walking with its simple linear formulation. The LIPM simplifies bipedal dynamics by assuming a point center of mass (CoM) for the main body with two massless legs. By assuming small vertical deviations of the center of mass, the plant dynamics become linear. We define the motion of the center of mass and these two feet in Cartesian space (x, y) as shown in Fig. (5.8A). The dynamics of the LIPM are

$$\dot{\mathbf{q}}_k = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{M} & \mathbf{0} \end{bmatrix} \mathbf{q}_k - \begin{bmatrix} \mathbf{0} \\ \mathbf{M} \end{bmatrix} \mathbf{u}_k; \quad \mathbf{M} = \begin{bmatrix} \frac{g}{z_0} & 0 \\ 0 & \frac{g}{z_0} \end{bmatrix} \tag{5.7}$$

where g is the gravity term, z_0 is the walking height, $\mathbf{q}$ is the predicted center of mass positions $\mathbf{p}$ & velocities $\mathbf{v}$, and $\mathbf{u}$ is the predicted center of pressure. Note $\mathbf{u}$ in (Eq. 5.7) is equivalent to the foot position since the LIPM assumes massless legs, a constant robot height (z_0), and point feet. Generally, the center of pressure position can be any point within a convex region around the foot position. We formulate LIPM motion planning as a convex MPC problem using reachability constraints and step length deviation cost as seen in [36, 90].

Reachability Constraints. A reachability constraint is included to ensure the planned footstep locations are kinematically attainable, and are defined for $h(\mathbf{q}_k, \mathbf{u}_k)$ in the following linear form:

$$\mathbf{s}_{\min} \leq (\mathbf{u}_k - \mathbf{p}_k) \leq \mathbf{s}_{\max} \tag{5.8}$$

where $\mathbf{s}_{\min}, \mathbf{s}_{\max}$ represent the maximum and minimum distance the feet can be away from the robot body.

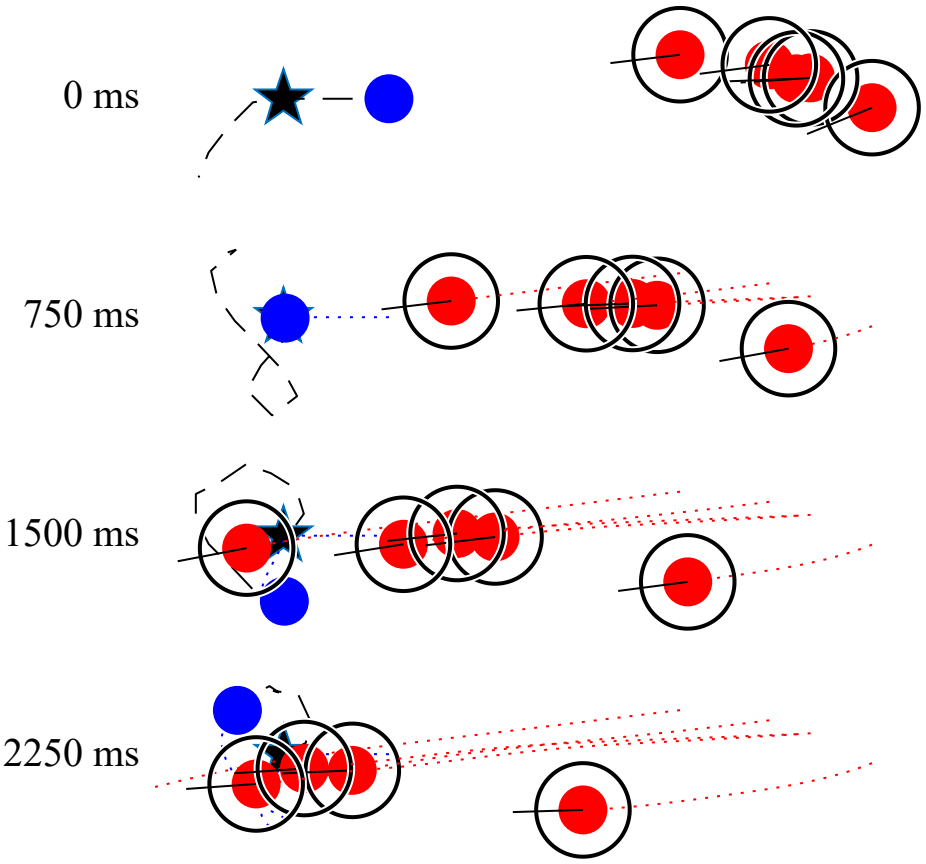

Figure 5.3: Simulated results of 2D uncoordinated waypoint tracking single agent (blue) with multi-adversary (red) avoidance. Despite the half-space constraints, the agent planned figure-eight looping maneuvers around adversaries. The four images show the tracking and avoidance trajectory through time (black) and the obstacle's traveled paths (red) over a 2.25 second period.

Step Length Cost. This cost function term is specific to the LIPM and minimizes the difference between the predicted step length $\mathbf{s}$ and the reference step length $\mathbf{s}_{\text{ref}}$. This drives the robot foot to the desired position from the current foot placement, and is formulated as such:

$$J_{\text{step}}(\mathbf{s}) = \sum_{k=1}^{N_s} ||\mathbf{s}_k - \mathbf{s}_{\text{ref},k}||^2_{\mathbf{W}_{\text{step}}} \tag{5.9}$$

where N_s is the number of robot steps predicted, and the number of MPC prediction steps is found using $N = N_s \frac{t_{\text{step}}}{dt}$. The frontal reference $s_{\text{ref},x}$ (or desired) step length is computed per:

$$s_{\text{ref},x} = v_{\text{ref},x}\ t_{\text{step}} \tag{5.10}$$

using the x directional walking speed $v_{\text{ref},x}$ and stepping time t_{step}. The sagittal step length $s_{\text{ref},y}$ is found using a nominal distance between the foot and robot CoM.

Based on the foot placement, we can compute the desired sagittal plane center of pressure reference trajectory $\mathbf{u}_{\text{ref},x} \in R^N$ for the J_{step} cost function term per:

$$\mathbf{u}_{ref,x} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ ... \\ ... \\ \mathbf{1} \end{bmatrix} p_{foot,x,1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & ... & \mathbf{0} \\ ... & ... & ... & ... \\ \mathbf{1} & \mathbf{1} & ... & \mathbf{1} \end{bmatrix} \begin{bmatrix} s_{x,1} \\ s_{x,2} \\ s_{x,3} \\ ... \\ s_{x,N_s} \end{bmatrix} \tag{5.11}$$

where $\mathbf{0}$ and $\mathbf{1}$ are column vectors with length $N_r = \frac{t_{\text{step}}}{dt}$. The x directional current foot placement is represented as $p_{\text{foot},x,1}$ and x directional predicted step length as s_x. The frontal plane center of pressure reference ($\mathbf{u}_{\text{ref},y}$) is found by substituting $p_{\text{foot},y,1}$ and $s_{x,1:N_s}$ in Eq. 5.11.

5.2.8 Cost Function

Although the weight matrix values $\mathbf{W}$ vary between models, components of the task cost function J_{task} can be generalized and are independent of the model. One common component in our examples minimizes the difference between the predicted states and the desired target state,

$$J_{\text{target}}(\mathbf{q}) = \sum_{k=1}^{N} ||\mathbf{q}_k - \mathbf{q}_{ref,k}||^2_{\mathbf{W}_{\text{target}}}, \tag{5.12}$$

where $\mathbf{q}_{ref}$ is the target state, with weighting matrix $\mathbf{W}_{\text{target}}$. For bipedal robots, the desired states $\mathbf{q}_{ref}$ are generated based on a commanded input (*e.g.*, stepping in place or walking forward with a certain velocity).

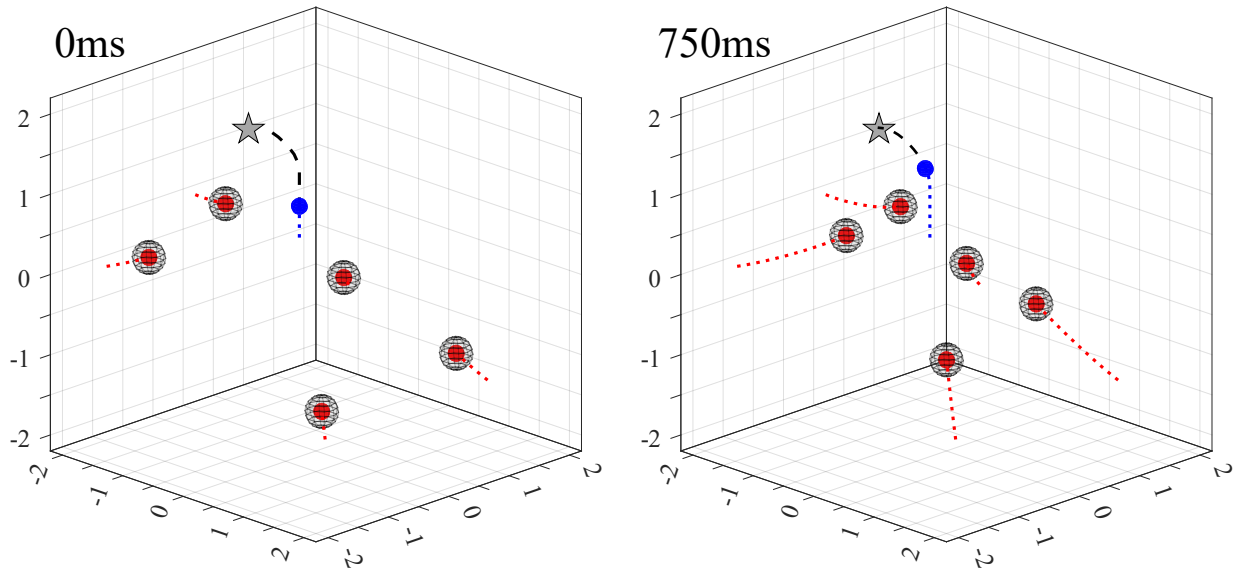

Figure 5.4: Simulated results of 3D uncoordinated waypoint tracking single agent (blue) with multi-adversary avoidance. Left and right images are sampled 0.75s apart.

Another common cost component minimizes the difference between control inputs and reference control inputs,

$$J_{\text{effort}}(\mathbf{u}) = \sum_{k=1}^{N} ||\mathbf{u}_k - \mathbf{u}_{ref,k}||^2_{\mathbf{W}_{\text{effort}}}. \tag{5.13}$$

For the double integrator model, this refers to minimizing the control inputs $\mathbf{u}$ and minimizing effort by assuming the reference input $\mathbf{u}_{ref}$ is zero. For the LIPM, this refers to minimizing the difference between the predicted control inputs $\mathbf{u}$ and the foot position reference $\mathbf{u}_{ref}$–encoding a preferred leg spread. In double-integrator demonstrations the cost function is $J = J_{\text{target}} + J_{\text{effort}}$, while the LIPM demonstrations include the stepping cost, $J = J_{\text{target}} + J_{\text{effort}} + J_{\text{step}}$.

5.3 Numerical Results

5.3.1 Simulated Environment

Simulations were run in MATLAB 2019b using the quadratic programming solver `quadprog` from the optimization toolbox. The constraints and cost function are pre-generated before the control loop using the `MatlabFunction` code generator and symbolic toolbox. After each elapsed sample period, the states are sampled, the planning optimization is solved, and the planned control inputs are applied to the system dynamics–which are subsequently integrated with MATLAB's `ode45` medium-order differential equation solver.

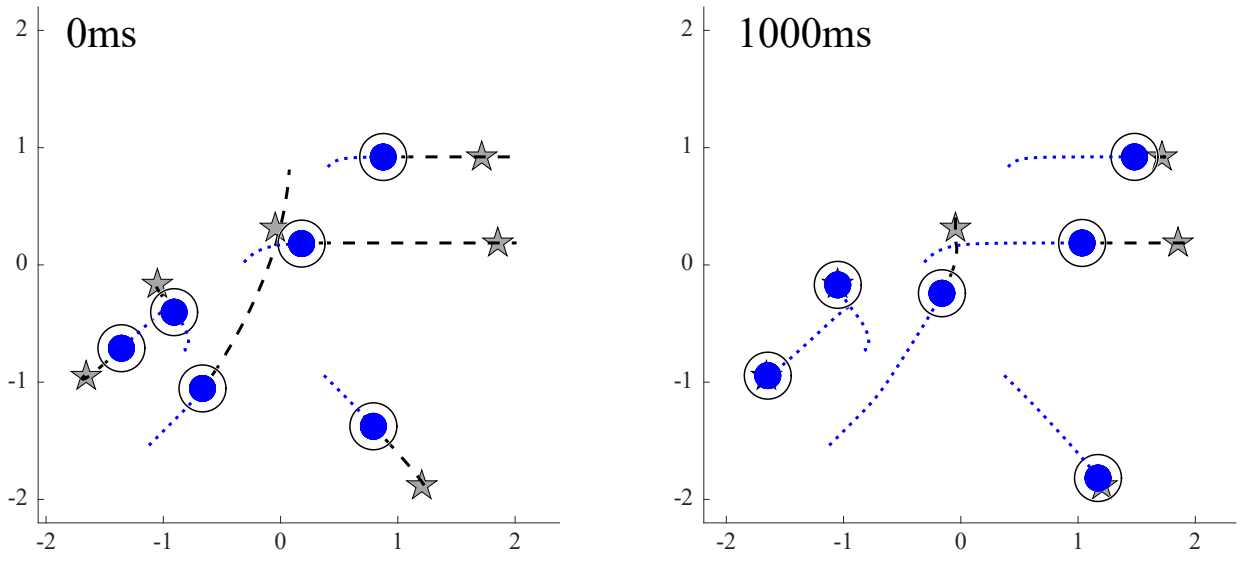

Figure 5.5: Simulated results of 2D uncoordinated multi-agent avoidance and waypoint tracking. Black dashed lines are motion plans. The left and right images were sampled 1 second apart.

A consistent 1.5sec time horizon is used with a 50ms sample time. This sample time is longer than necessary given a typical optimization required only $\sim$ 7ms of computation time. Recent testing has further decreased this time to $\sim$ 2ms (500Hz) using the OSQP solver [84].

5.3.2 Dynamic Obstacle

Early testing of the avoidance setup included an obstacle moving at a constant acceleration across the agent's desired target point. The agent then plans an avoidance maneuver with a trajectory leading back to the desired state. Originally, we simply used a half-plane constraint to represent the obstacle Fig. 5.2A. However, linear estimations of the dynamics and discrete time intervals commonly resulted in avoidance constraint violations and thus infeasibilities. We found adding the soft avoidance constraints to be an effective method of mitigating optimization failures– drastically improving obstacle avoidance. Specifically, it allowed previous optimizations to solve reliably, as depicted in Fig. 5.2D. We further explore similar single-obstacle scenarios on drone hardware in Sec. 5.4.

5.3.3 Adversarial Pursuit

We expand the previous scenario by altering control of the obstacle(s) to be adversarial. Using a basic PD control law, the adversary(s) (or obstacle(s)) track and chase the agent. The agent is

tasked to go to a target point which is randomly moved to a new position every 4 seconds. The dynamics for both systems are nearly identical (Eq. 5.6), with the adversary(s) having a small damping term to reduce overshoot when tracking the agent. Again, these are non-cooperative entities, and the agent has no information about the adversary's intent other than its current position and velocity. We find even when we limit the agent's velocity to be substantially lower than the adversary's velocity, the agent successfully avoids the adversary.

2D Multi-obstacle. Here we increase the complexity by avoiding 5 adversaries, each with randomized gains and starting positions Fig. 5.3. In some situations, it would appear the agent (in blue) was trapped, but this simple heuristic method would find routes that encroached on the adversary's (in red) soft constraints (black circle). Sec. 5.3.5 quantifies the performance reliability with up to 10 adversaries, but we were able to demonstrate successful evasion with up to 15, the largest number tested.

3D Simulation. We further extend the method to three dimensions and find similar results Fig. 5.4. Using constraint planes to navigate around obstacles we demonstrate successful scaling to 3D in both performance and tractable computation.

5.3.4 Multiple Agents (Go Home)

We now demonstrate collision avoidance among multiple MPC agents that are neither adversarial nor cooperative. We simulate multiple agents, independently optimizing their motion plans to reach target points and assess their performance when their planned paths conflict.

2D Multiple Agents. Here we model 6 systems, all planning agents, in a 2D environment and randomize new target points every 4 seconds. We again find similar results Fig. 5.5 to the obstacle avoidance, where this heuristic setup is robust to a dynamic environment, and the agents do not collide with each other. The setup presented here uses individual QPs solved for each agent's trajectory.

A common mode of failure is the "who goes first?" conundrum–where two agents moving in parallel need to cross each other such that one must speed up or slow down (*i.e.*, a stalemate). This was particularly problematic in earlier formulations that did not iteratively update half-space cuts along previously solved trajectories. Including the iterative half-space update anecdotally reduced the incidence of these stalemates. Although our multi-agent results are independent optimizations,

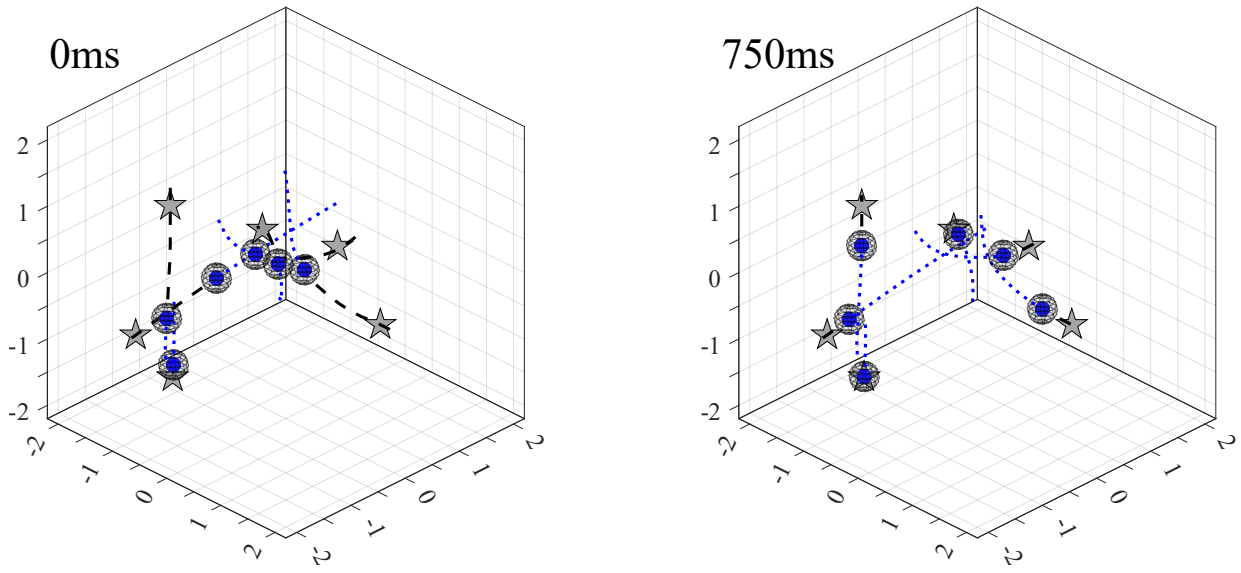

Figure 5.6: Simulated results of 3D uncoordinated multi-agent avoidance and waypoint tracking. The left image shows the initial plan to waypoints with black dashed lines, and the right shows 0.75 seconds into the simulation.

we could generate cooperative swarming at the cost of longer computation times (*i.e.*, multi-agent MPC) by combining all planning operations into a single coupled optimization.

3D Multiple Agents. We further extend the method to three dimensions and find similar results as seen in Fig. 5.6B, again demonstrating the proposed method scales effectively to 3D for multiple non-cooperative agents.

5.3.5 Reliability Results

To understand the effect of our soft constraint, we simulate the 2D system starting with no soft penalty and ending with a soft penalty 1.5 times the $\mathbf{d}_{obs}$ for 1 through 10 obstacles.

$$0 \geq \mathbf{d}_k \cdot (\tilde{\mathbf{d}}_k - \mathbf{p}_k) - \delta_k \qquad \text{(Soft Constraint)}$$

$$\tilde{\mathbf{d}}_k = (d_{\text{agent}} + d_{\text{obst}} + d_{\text{risk}})\hat{\mathbf{d}}_k$$

$$d_{\text{risk}} = \mathbf{j}\ d_{\text{obst}} \qquad \mathbf{j} = \{0, 0.3, ..., 1.5\}$$

Starting at a position of $\mathbf{p}_{\text{agent},1} = [4, 0]^T$, the agent is tasked to travel to $\mathbf{v}_{\text{agent},1} = [0, 0]^T$, with adversarial obstacles placed in its path. Initial obstacle locations are selected at random within a range of $1 \leq x \leq 3$ and $-0.5 \leq y \leq 0.5$. Each obstacle uses a PD control law with randomized gains to "chase" the agent. Each simulation runs until the agent either reaches the goal with a

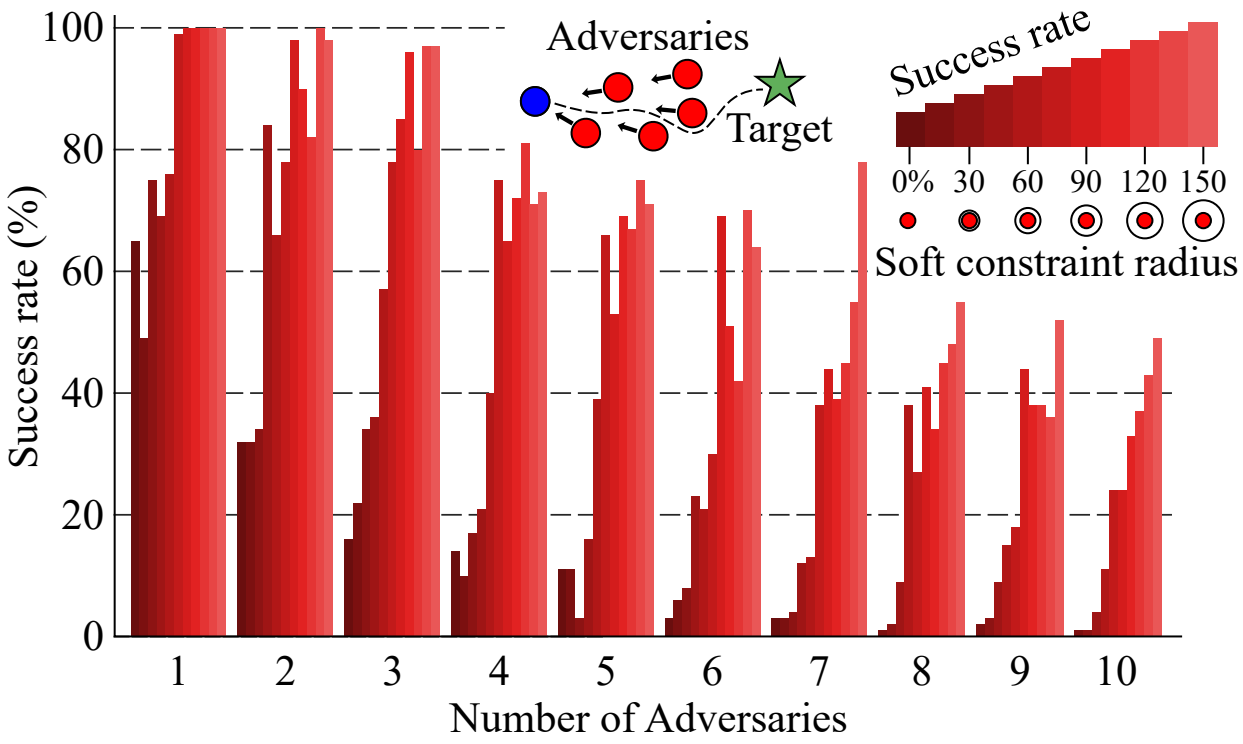

Figure 5.7: Reliability testing of multiagent avoidance in a tight corridor as a function of soft constraint radius.

velocity near zero or collides with an obstacle. The corresponding success rates out of 100 trials are presented in Fig. 5.7.

5.3.6 Legged Locomotion

In this section, we will show that the obstacle avoidance algorithm can be easily adapted for other robotics applications like legged robot motion generation.

Here we show the LIPM-MPC simulation result Fig. 5.8A. The robot stepping time t_{step} is 0.5s with a sample time dt of 0.1s. The motion planner predicts $N_s = 4$ steps into the future and we command the robot to simply walk forward. The simulated results without obstacle avoidance Fig. 5.8B and with obstacle avoidance Fig. 5.8C show that the avoidance algorithm can guide the robot to walk around the obstacle safely. Further, we can tune the MPC soft penalty terms and add more points to the robot for collision detection (*e.g.*, robot feet, knees, etc.) to extend the LIPM-MPC capabilities for more complex geometries and dynamic environments.

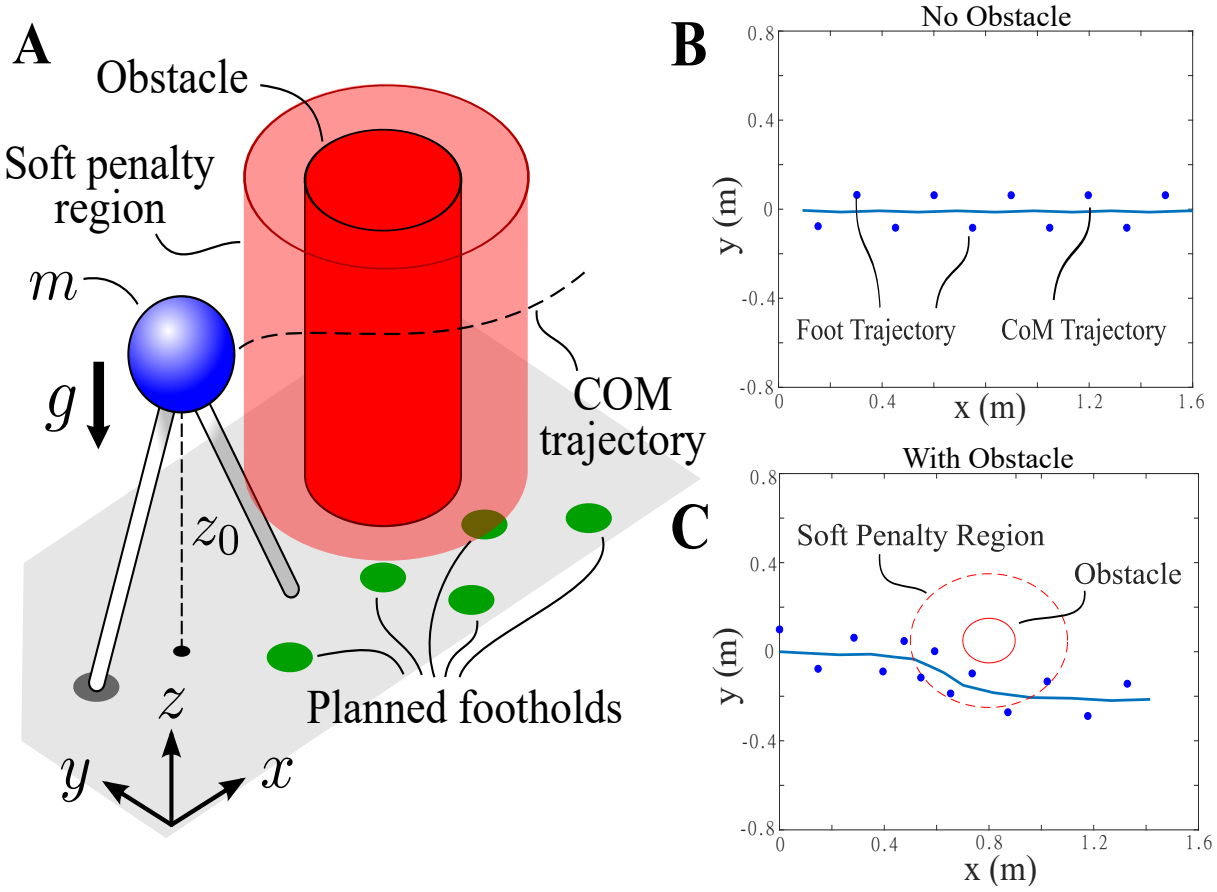

Figure 5.8: Bipedal motion planning in real-time using a Linear Inverted Pendulum Model (LIPM). **A.** Diagram for the numerical experimental setup. **B.** An MPC-generated motion without an obstacle. **C.** Real-time bipedal obstacle avoidance using the presented motion planner.

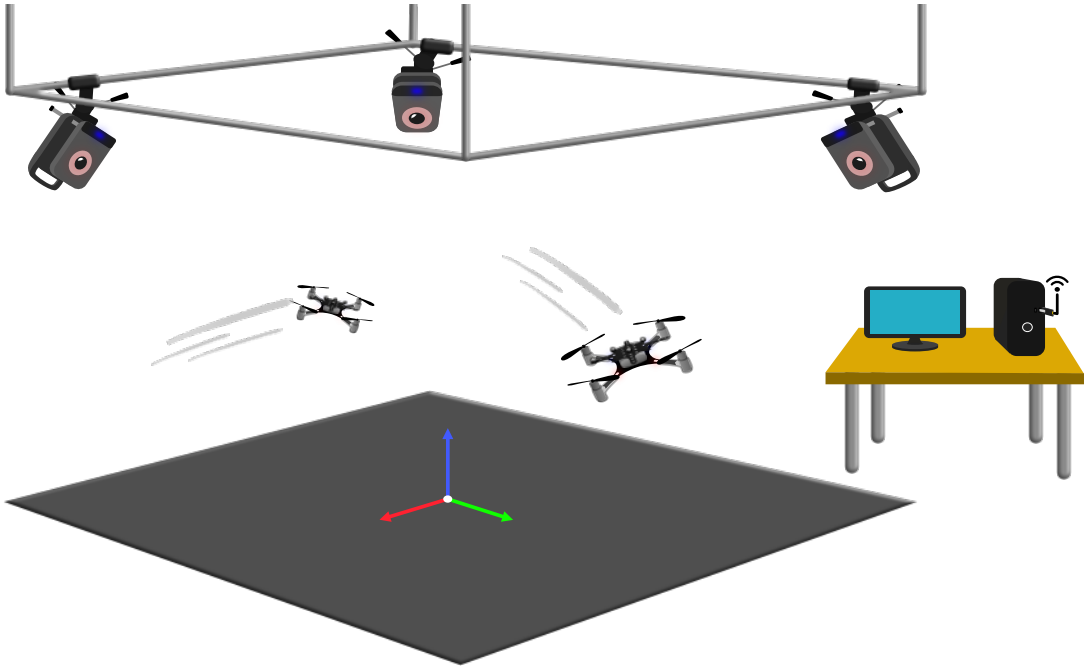

Figure 5.9: Lab experimental setup for quadrotors. Vicon cameras track the location of all agents and adversaries to provide sensory feedback.

5.4 Hardware Experiments

5.4.1 Experimental Setup

Hardware Description. Hardware experiments were carried out using a BitCraze quadrotor. The BitCraze quadrotor CrazyFlie uses brushed DC motors and is controlled from a Dell XPS 8700 with an Intel i7-4790 processor and 14GB of RAM base computer using a 2.4Ghz radio USB dongle. For tracking the quadrotor's state, we use a Vicon motion capture system. Our Vicon setup uses eight infrared cameras to track the position of reflective markers and measures drone states with a sub-millimeter precision at an approximate sampling rate of 150Hz.

Software. BitCraze uses open-source code which runs a low-level control loop at 1kHz and has been extended/modified by several other universities. The University of Southern California developed a software package called "Crazyswarm" [71] that uses a ROS node to control a swarm of up to 50 drones. We use this code base because of its motion capture integration, simulation environment, and user-friendly Python-based firmware. Of the available motor level controllers (PID, INDI, or Mellinger) from Crazyswarm, we use the Mellinger controller [61]. It allows for steep roll/pitch angles and develops minimum snap trajectories in real-time using a sequence of

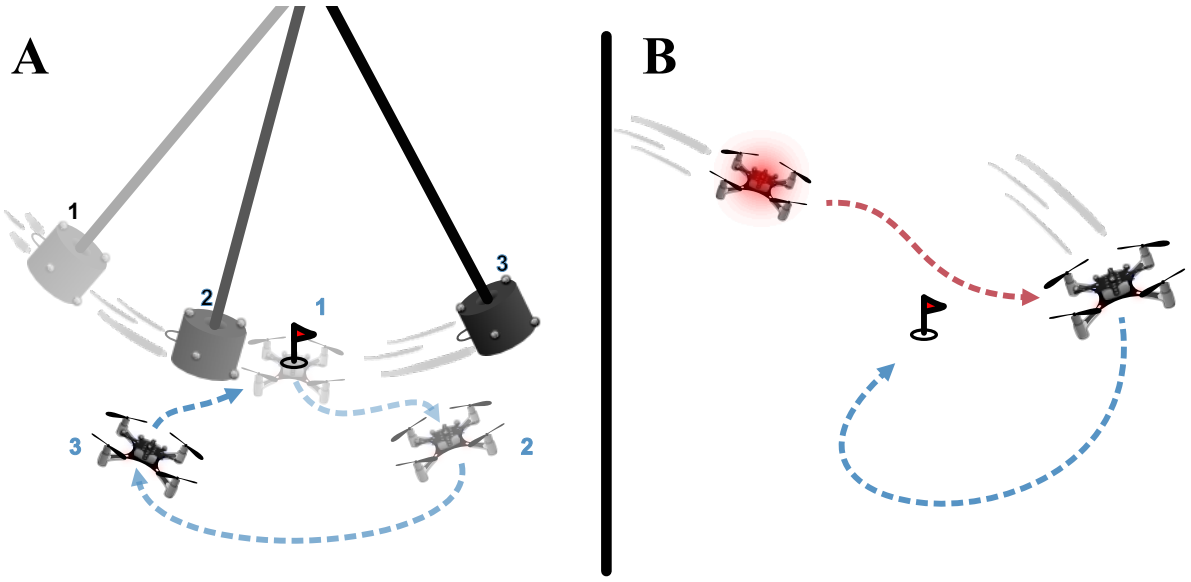

Figure 5.10: In our experiments, the quadrotor must minimize its distance to a waypoint while **A.** avoid a swinging pendulum and **B.** avoiding a chasing adversary quadrotor controlled by a PD chase law.

desired Cartesian positions $[x\ y\ z]$, yaw angle (ψ), and their subsequent derivatives. We include our avoidance method in the repository by calling MATLAB from Python and writing a `.csv` file from the trajectory points planned by MPC. The main execution control loop runs at 20Hz using points from the most recent trajectory.

5.4.2 Systematic Experiment: Pendulum Avoidance

To test the effectiveness of the "at risk" soft distance constraint, we incrementally changed the d_{risk} value for the quadrotor and proceeded to swing a pendulum at it (3 times for each of the 7 radii), as seen in Fig. 5.10A. The 0.2kg pendulum bob was encased in foam and hung by fishing line 3.2m below the ceiling. To begin the test, the pendulum is set at a fixed height on a post and held using a pin-release mechanism. The drone is placed in the motion capture volume and commanded to hover 0.3m above the origin so that the pendulum is sure to strike it. For each experiment, the pendulum was released from the same height and the drone was commanded to hold the same desired state. As seen in Fig. 5.11, a $d_{risk} \geq 15$cm worked every time–even with the pendulum released from the highest possible height, **reaching speeds greater than 10m/s**.

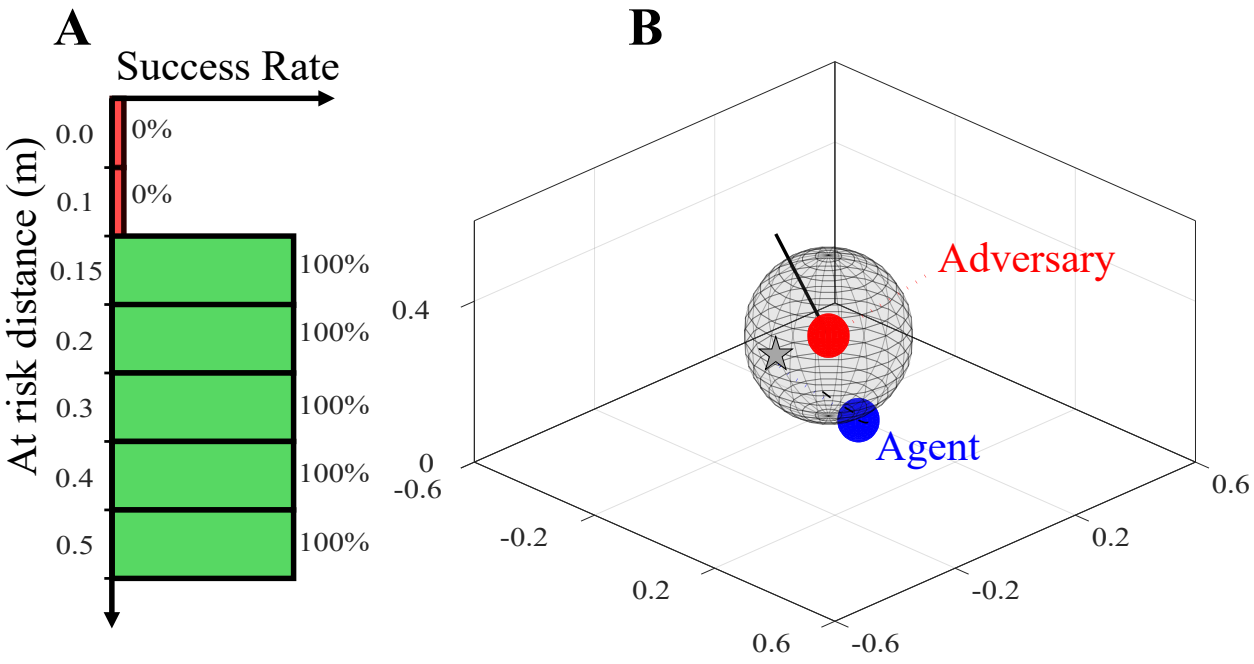

Figure 5.11: Pendulum experiment results. **A.** Soft penalty distance, d_{risk} and the avoidance success rate for 3 pendulum swing attempts. **B.** Example trajectory data from pendulum test.

5.4.3 Demonstration: Adversarial Pursuit

We validate the numerical results from Section 5.3 by mimicking the adversary setup. As seen in Fig. 5.9B, two CrazyFlie quadrotors are placed apart from each other at random positions within the motion capture field. The agent uses the MPC avoidance method, the adversary chases the agent position using a PD controller, and both use a Mellinger controller for low-level control.

2D Adversary Avoidance. Our initial adversarial experiments were constrained the avoidance to two dimensions. Representative experimental data can be seen in Fig. 5.12B where the adversary approaches the agent, causing the agent to plan a trajectory around the adversary, temporarily leaving its target point to dodge.

3D Adversary Avoidance. We extended our experiments into three dimensions as seen in the experimental results in Fig. 5.12C. Similar to the 2D adversary avoidance tests, the agent was able to avoid the chasing adversary. One notable effect was the down-wash disturbance when one drone passed below the other, which would cause instability in the PD-controlled chase drone.

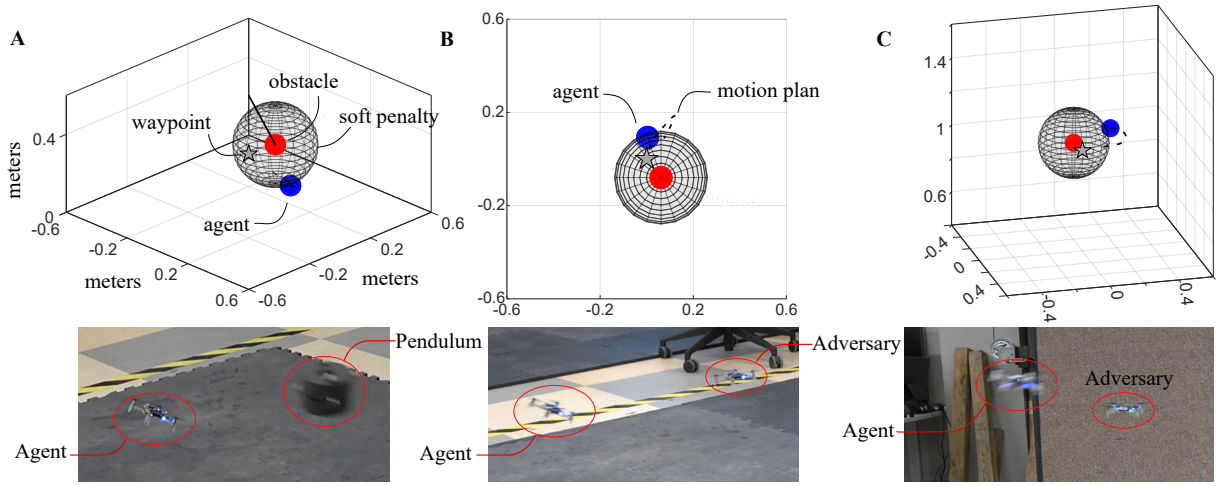

Figure 5.12: Experimental results trajectory data and example images. **A.** Quadrotor avoids a pendulum swinging at speeds up to 10m/s. **B.** The quadrotor is chased by an adversary drone it must avoid while staying within a 2D horizontal plane. **C.** Quadrotor adversary chase in 3D. See supplementary video for real-time footage.

5.5 Discussion

Avoidance Behavior. There was a clear correlation between the slack variable weight $\mathbf{W}_{\text{avoid}}$, the soft constraint distance d_{risk}, and the avoidance performance. When the weight was low, the agent would favor passing the soft constraint, commonly resulting in a collision. Alternatively, a small soft constraint distance with a very large weight resulted in more reliable obstacle avoidance. Overall we find that making the slack variable weight significantly larger than the other cost function weights to be the most effective method of enforcing avoidance.

Although methods for automatically tuning these parameters are beyond the scope of this paper, we believe that developing these values depends on both the obstacle's/agent's dynamically constrained maneuverability relative to each other and the obstacle's/agent's size. We realize the values at each node could be chosen using a more methodical manner, but the results presented here simply use a scalar value across all nodes for the soft constraint penalty and soft constraint distance. Despite this simplification, the agents, both simulated and physical, can successfully avoid dynamic obstacles, which we believe speaks to the efficacy of the proposed half-space method with soft constraints.

5.5.1 Future Work

avoidance capability of the motion planner on hardware is promising for extensions to the methods detailed here. The half-space approach can be extended by further developing predicted trajectory $\mathbf{p}^*_{\text{obst},k}$ or by developing non-spherical soft constraints using velocities and accelerations for a more complex d_k. Recent work has already accomplished this by including elliptical obstacles for a bipedal robot. Including vision should also allow for oddly shaped polygons and corridors by sensing only the closest points of collision on an obstacle.

While the method was robust to poor modeling assumptions of adversaries, future work could seek to model them on-the-fly for improved performance. Further, while the 500Hz control rate was sufficient for real-time, we believe straightforward software changes could significantly raise this rate and avail these methods to hardware platforms with faster dynamics.

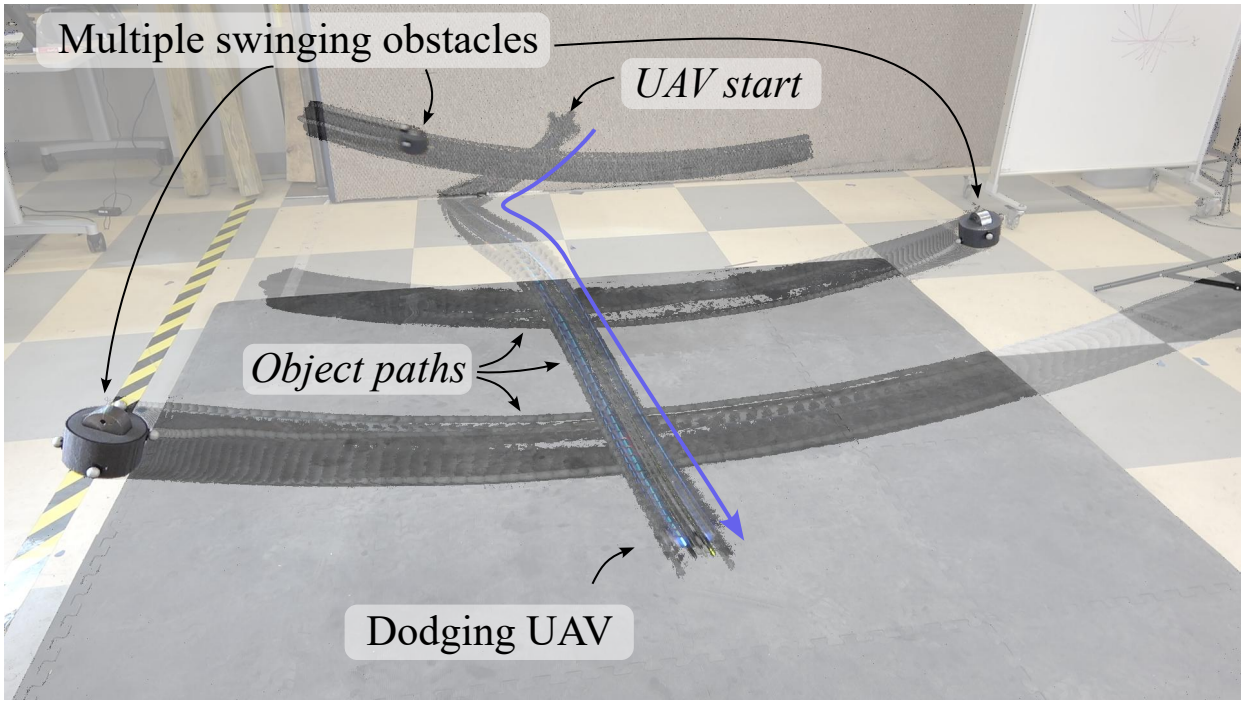

Figure 5.13: Stroboscopic image of a quadrotor UAV avoiding multiple swinging obstacles in its path.

5.6 Conclusion

We presented a motion planning method that allows for cross-platform real-time planning while avoiding multiple uncooperative and adversarial obstacles in 2D and 3D environments. Even when

the obstacles are adversarial or move at high speeds, the agent safely navigates around them in both simulation and hardware. This method was also applied to a bipedal walking model avoiding a stationary obstacle - demonstrating applicability to varied locomotion modes. We validated these methods on a quadrotor, tasking it to (1) avoid a 10m/s pendulum swing and (2) avoid a chasing quadrotor adversary.

CHAPTER 6

CONCLUSION

This thesis contributed and tested robot control algorithms for applications ranging from obstacle–avoidance with unmanned aerial vehicles, to legged locomotion through viscous media, and full-scale humanoid balancing. They do however, all come to a common consensus:

1. Designing controllers for dynamic scenarios can improve a system's performance and reliability
2. Real-time re-planning and control is an effective method for maintaining stability, efficiency, and agility in these dynamic environments.

6.1 Key Chapter Conclusion

The following are the key findings of the research presented in this dissertation.

- **Chapter 2:** Large, high degree of freedom systems can be controlled on dynamic terrain using simple models and high control rates.
- **Chapter 3:** Trajectories for a monopod can be generated via convex optimizations without using predefined contact timings, sequences, or locations. The resulting behaviors, such as hopping forward or standing still, emerge from the general optimization formulation.
- **Chapter 4:** Simplified estimates of viscous media dynamics can be used with nonlinear optimizations to control a monopod in both simulation and hardware.
- **Chapter 5:** Soft piecewise constraints and half-space obstacle representation can be used in a convex optimization to rapidly re-plan and avoid at least ten adversarial obstacles. This method works across locomotion modes in simulated environments and on hardware.

6.2 Discussion

6.2.1 The Future of Model-Based Optimizations

All the research presented in this thesis directly operates on dynamic system models, many of which are included in optimizations to generate motion plans. These "model-based optimizations" have been the dominant method of control until recently when significant progress has been made via reinforcement learning (RL) approaches to training control policies. The results seen in the

legged locomotion community (*e.g.*, ETH's ANYmal [58]) using this methodology are compelling and beg the question, is there even a place for model-based optimizations in the future? Generally, this author believes there is because of aspects where model-based methods outperform RL:

1. Optimizations are computationally efficient and reliable for convex problems where globally optimal solutions can be solved at kHz control rates.
2. Both the problem and objective are well-defined (with no intermediary black box) so control theorists have a high level of intuition about how the objective function weights affect the generated solution. This allows for transparency into how the optimal solution was developed and makes behavior-shaping and debugging issues more straightforward.

Alternatively, RL outperforms model-based methods in:

1. The ability to improve over time since it learns from simulated experiences. Exposing a system to a wide variety of scenarios and perturbations allows the system to develop a larger database of optimal reactions, thus improving the controller's robustness.
2. Its handling of complex and unseen environments. The volume of scenarios a controller is exposed to means there is no need for a high-fidelity model and it's able to handle stochastic environments by simply defining a goal.

In the future, this author believes the most successful controllers will find a balance between these model-based and model-free control schemes. RL is best implemented in situations where dynamics and tasks are difficult to model tasks whereas model-based optimizations are advantageous when real-time solutions for novel scenarios are necessary.

6.2.2 Chapter 2: Lessons from Force-Controlled Balancing

Bipedal controllers can remain stable in the face of unplanned contact changes (*e.g.*, soft, moving, or completely lost footholds) if the system's joints can be torque controlled, can move quickly, and are not damaged from large impacts. This applies to systems that are designed with series elastic actuation and low-impedance drives. This style of control is especially useful for systems where the dynamic model may be inaccurate, unknown, or changing. The controller's success can be attributed to the lack of reliance on a complex dynamic model. By relying only on a point mass model and kinematic Jacobians, the author was able to produce a stable force controller that is invariant to contact changes. The results show a contact invariant bipedal controller that works on dynamic terrain can be developed without using foothold detection.

Viability of Force Control: The author first presents a bipedal balancing force controller implemented on hardware. This style of control is effective for dynamic terrain because the controller focuses on controlling end effector forces rather than positions. To this author's knowledge, this is the first example of force control implemented on the Cassie bipedal platform, and the author shows that the approach works well in numerous scenarios even with a minimalist dynamic model. The viability of this control method for future work is further supported by the research published with both TORO [1] and MIT's Cheetah 3 [24] which use force control and couple it with higher-level planners. These bipedal and quadrupedal platforms display impressive results showcasing the system on a wide variety of stationary, soft, and moving terrains. One common drawback with this style of control is the hardware requirements. To use force control on a platform, the robot must be able to effectively render torques at the outputs of the joints. Unfortunately, many existing robots are unable to utilize this style of control because this is generally accomplished during the design phase either by incorporating low-friction gearboxes or direct torque measurement at all joints.

6.2.3 Chapter 3: Lessons from Through Contact Motion Planning

Monopod motion plans without predefined footfall locations or timing can be generated in milliseconds by solving quadratic programs. This method allows the system to hop, stand, or transition between the two at a variety of velocities and heights by only changing the desired speed and body height. The author was able to accomplish this by using a modified version of SLIP dynamics and exploiting the inevitability of a footfall contact. By understanding at some point the foot (or body) must hit the ground, the author predicts the mode schedule and an associated linear approximation of the dynamics at each time point. There has been no prior published work combining these two aspects which allow for a convex formulation of the through-contact problem for monopods.

Minimizing User-Defined Gait Parameters: The author presents a method of monopod control that uses few predetermined control parameters. Most planning-based legged robot controllers use predefined step timing, gait sequence, and/or foot placements. The controllers that don't (such as linear complementarity problems [87]), tend to be slow due to the inherent nonlinearities associated with leaving these parameters as free variables. The monopod presented here removes these predefined aspects and maintains high control rates by linearizing the dynamics and modifying the constraints to exploit "inevitable" robot states in legged locomotion (*i.e.*, foot eventually hitting the ground). The author realizes that the work presented here is limited to two proof-of-concept

monopod models. When this formulation was preliminarily extended to a two-legged model, the controller only utilized one leg to hop around rather than planning a multi-legged contact sequence. This research laid the groundwork as a proof of concept for this style of control, but further research is needed to develop methods in which multi-legged systems make use of additional legs while maintaining optimization convexity and fast solve times.

6.2.4 Chapter 4: Lessons Learned from Motion Planning Through Fluidic Media

Partially submerged fluid forces can be estimated as a differentiable function to allow for fast-solving optimizations. When these forces are included in the optimization as integrals, the optimization takes several minutes to solve, but formulating the integral as a "smooth" function can reduce that to seconds. This is accomplished by lumping the integral into a single force which allows the function to become continuously differentiable. Other methods of legged locomotion through fluid-like media fail to generate gaits in a feasible time scale for online operation or do not consider the cost of transport in the motion plan. The method presented here is a more practical method to generate the gaits quickly, thus potentially enabling real-time control in a changing resistive environment.

Why Model Environmental Factors: The author discusses a framework for optimizing trajectories of a reduced-order legged model running through resistive media during operation ($\sim 0.5Hz$) and tests it on a two-degree-of-freedom "Minitaur" leg. Previous work synthesizing legged locomotion through viscous media tends to be computationally slow (on the order of minutes [3], [7]) and thus impractical for real-time feedback control. This work balances modeling accuracy and control rates by reducing the model but keeping control rates between $1-2$ sec. This means were can plan a new trajectory every $2-3$ gait cycles. Modeling these environmental factors is important for accurate control and minimizing the energetic cost of transport. Ideally, we want systems that adapt to their environment while maintaining locomotion efficiency as well as the speed and reliability of gait computation. This work is a step toward that goal where the author kept the problem computationally tractable but continued to model the unique dynamic nonlinearities. In the future, the author would like to see the research extended to include on-the-fly identification of media properties so the model can update the fluid-dynamic properties of the media, yielding sub-second adaptation to a changing dynamic environment.

6.2.5 Chapter 5: Lessons Learned from Dynamic Obstacle Avoidance

Motion plans with obstacle avoidance included can be formulated in real-time for multiple fast dynamic obstacles on hardware with varied modes of locomotion using the algorithm presented. The mathematical formulation is straightforward, the weights easy to tune, and the overall setup is easy to implement. Using three key components the obstacle avoidance algorithm is used on a micro UAV and biped to avoid fast-moving dynamic obstacles. First, the author includes a slack variable that penalizes the objective when the agent and obstacle have close proximity. Second, the author iteratively linearizes the dynamics along the trajectory using the most recently generated motion plan. Finally, the author estimates the obstacle's motion plan along our time horizon using a first-order integration of the measured states. Prior publications do not combine these three methods; however, the simulated and hardware results show how effective the method is. To our knowledge, this is the first obstacle avoidance algorithm that works with highly dynamic drones and bipedal robots for many dynamic and adversarial obstacles.

How to Avoid Moving Obstacles: Finally, the author presented a motion planning method that allows for cross-platform real-time planning while avoiding multiple uncooperative and adversarial obstacles in 2D and 3D environments. Previous methods of avoidance for uncontrolled moving obstacles find motion plans using linear algebra techniques, convex programs, or nonlinear programs. What differentiates this research from previous methods is the use of piecewise soft constraints in a convex problem that has been analyzed using a parameter study, and implemented on hardware platforms with varied modes of locomotion. To the author's knowledge, it is the only method shown to work with multiple locomotion modes for fast-moving adversarial obstacles and offers an alternative to control barrier functions and velocity obstacle methods. Two pitfalls still exist with the presented methods: (1) the lack of guarantee that an obstacle will be avoided, and (2) the current problem formulation which only allows for circular or spherical obstacles. This author believes that with further development both of these issues can be resolved to both guarantee avoidance (if at all possible), and include more diverse shapes such as polygons, corridors, and other shapes.

BIBLIOGRAPHY

[1] Firas Abi-Farraj et al. "Torque-based balancing for a humanoid robot performing high-force interaction tasks". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2023–2030.

[2] Mojtaba Ahmadi and Martin Buehler. "Controlled passive dynamic running experiments with the ARL-monopod II". In: *IEEE Transactions on Robotics* 22.5 (2006), pp. 974–986.

[3] Ryan Alicea, Kyle Ladyko, and Jonathan Clark. "Lift your leg: Mechanics of running through fluids". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7455–7461.

[4] Aaron D Ames et al. "Control barrier function based quadratic programs for safety critical systems". In: *IEEE Transactions on Automatic Control* 62.8 (2016), pp. 3861–3876.

[5] Ben Andrews et al. "Running over unknown rough terrain with a one-legged planar robot". In: *Bioinspiration & biomimetics* 6.2 (2011), p. 026009.

[6] Taylor Apgar et al. "Fast Online Trajectory Optimization for the Bipedal Robot Cassie." In: *Robotics: Science and Systems*. Vol. 101. Pittsburgh, Pennsylvania, USA. 2018, p. 14.

[7] Max P Austin and Jonathan E Clark. "The Fluid Field SLIP Model: Terrestrial-Aquatic Dynamic Legged Locomotion". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4948–4954.

[8] Alp Aydinoglu and Michael Posa. "Real-time multi-contact model predictive control via admm". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 3414–3421.

[9] Tomas Baca et al. "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 6753–6760.

[10] Moses Bangura and Robert Mahony. "Real-time model predictive control for quadrotors". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 11773–11780.

[11] Daman Bareiss and Jur Van Den Berg. "Generalized reciprocal collision avoidance". In: *The International Journal of Robotics Research* 34.12 (2015), pp. 1501–1514.

[12] Jur Van Den Berg et al. "Reciprocal n-Body Collision Avoidance". In: *Robotics research*. Springer, 2011, pp. 3–19.

[13] Lorenz T Biegler and Victor M Zavala. "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization". In: *Computers & Chemical Engineering* 33.3 (2009), pp. 575–582.

[14] Robert Bogue. "Domestic robots: Has their time finally come?" In: *Industrial Robot: An International Journal* 44.2 (2017), pp. 129–136.

[15] Will Bosworth et al. "Robot locomotion on hard and soft ground: Measuring stability and ground properties in-situ". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 3582–3589.

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

[17] Marcello Calisti and Cecilia Laschi. "Underwater running on uneven terrain". In: *OCEANS 2015-Genova*. IEEE. 2015, pp. 1–5.

[18] Alexander H Chang et al. "Every Hop is an Opportunity: Quickly Classifying and Adapting to Terrain During Targeted Hopping". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3188–3194.

[19] Iordanis Chatzinikolaidis and Zhibin Li. "Trajectory optimization of contact-rich motions using implicit differential dynamic programming". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2626–2633.

[20] Yongbo Chen, Shoudong Huang, and Robert Fitch. "Active SLAM for mobile robots with area coverage and obstacle avoidance". In: *IEEE/ASME Transactions on Mechatronics* 25.3 (2020), pp. 1182–1192.

[21] Hui Cheng et al. "Decentralized navigation of multiple agents based on ORCA and model predictive control". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 3446–3451.

[22] Simon Le Cleac'h et al. "Fast contact-implicit model-predictive control". In: *arXiv preprint arXiv:2107.05616* (2021).

[23] Xingye Da and Jessy Grizzle. "Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots". In: *The International Journal of Robotics Research* 38.9 (2019), pp. 1063–1097.

[24] Jared Di Carlo et al. "Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–9.

[25] Yanran Ding, Chuanzheng Li, and Hae-Won Park. "Kinodynamic motion planning for multi-legged robot jumping via mixed-integer convex program". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 3998–4005.

[26] Lefteris Doitsidis et al. "A framework for fuzzy logic based UAV navigation and control". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 4. IEEE. 2004, pp. 4041–4046.

[27] Johannes Englsberger et al. “Overview of the torque-controlled humanoid robot TORO”. In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2014, pp. 916–923.

[28] Michael Ernst, Hartmut Geyer, and Reinhard Blickhan. “Extension and customization of self-stability control in compliant legged systems”. In: *Bioinspiration & biomimetics* 7.4 (2012), p. 046002.

[29] Siyuan Feng et al. “Optimization based controller design and implementation for the ATLAS robot in the darpa robotics challenge finals”. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2015, pp. 1028–1035.

[30] Paolo Fiorini and Zvi Shiller. “Motion planning in dynamic environments using velocity obstacles”. In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[31] Wei Gao et al. “Fast, versatile, and open-loop stable running behaviors with proprioceptive-only sensing using model-based optimization”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 483–489.

[32] Grant Gibson et al. “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)*. IEEE. 2022, pp. 6724–6731.

[33] Yukai Gong et al. “Feedback Control of a Cassie Bipedal Robot: Walking, Standing, and Riding a Segway”. In: *Proceedings of the American Control Conference*. 2019.

[34] Duncan W Haldane et al. “Robotic vertical jumping agility via series-elastic power modulation”. In: *Science Robotics* 1.1 (2016), eaag2048.

[35] Bernd Henze, Máximo A Roa, and Christian Ott. “Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios”. In: *The International Journal of Robotics Research* 35.12 (2016), pp. 1522–1543.

[36] Andrei Herdt et al. “Online Walking Motion Generation with Automatic Footstep Placement”. In: *Advanced Robotics* 24.5-6 (2010), pp. 719–737. DOI: 10.1163/016918610X493552.

[37] Ayonga Hereid, Shishir Kolathaya, and Aaron D Ames. “Online optimal gait generation for bipedal walking robots using legendre pseudospectral optimization”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 6173–6179.

[38] Ayonga Hereid et al. “Dynamic humanoid locomotion: A scalable formulation for HZD gait optimization”. In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 370–387.

[39] Christian Hubicki et al. “Do limit cycles matter in the long run? stable orbits and sliding-mass dynamics emerge in task-optimal locomotion”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 5113–5120.

[40] Christian Hubicki et al. “Walking and Running with Passive Compliance: Lessons from Engineering: A Live Demonstration of the ATRIAS Biped”. In: *IEEE Robotics & Automation Magazine* 25.3 (2018), pp. 23–39.

[41] Christian Hubicki et al. “Walking and Running with Passive Compliance: Lessons from Engineering: A Live Demonstration of the ATRIAS Biped”. In: *IEEE Robotics Automation Magazine* 25.3 (2018), pp. 23–39. DOI: 10.1109/MRA.2017.2783922.

[42] Koji Ishihara, Takeshi D Itoh, and Jun Morimoto. “Full-body optimal control toward versatile and agile behaviors in a humanoid robot”. In: *IEEE Robotics and Automation Letters* 5.1 (2019), pp. 119–126.

[43] Fabian Jenelten et al. “Dynamic Locomotion on Slippery Ground”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4170–4176.

[44] Mikhail S Jones. “Optimal control of an underactuated bipedal robot”. In: (2014).

[45] S. Kajita et al. “Biped walking pattern generation by using preview control of zero-moment point”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. 2003, 1620–1626 vol.2. DOI: 10.1109/ROBOT.2003.1241826.

[46] S. Kajita et al. “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*. Vol. 1. 2001, pp. 239–246. DOI: 10.1109/IROS.2001.973365.

[47] Shuuji Kajita et al. “Biped walking pattern generation by using preview control of zero-moment point”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. Vol. 2. IEEE. 2003, pp. 1620–1626.

[48] Mina Kamel et al. “Nonlinear model predictive control for multi-micro aerial vehicle robust collision avoidance”. In: *arXiv preprint arXiv:1703.01164* (2017).

[49] Hyun-jin Kang et al. “Biped walking stabilization on soft ground based on gait analysis”. In: *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE. 2012, pp. 669–674.

[50] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. “Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 6295–6301. DOI: 10.1109/ICRA.2019.8793865.

[51] Oussama Khatib. “A unified approach for motion and force control of robot manipulators: The operational space formulation”. In: *IEEE Journal on Robotics and Automation* 3.1 (1987), pp. 43–53.

[52] Donghyun Kim et al. “Control Scheme and Uncertainty Considerations for Dynamic Balancing of Passive-Ankled Bipeds and Full Humanoids”. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 1–9.

[53] Donghyun Kim et al. “Stabilizing series-elastic point-foot bipeds using whole-body operational space control”. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1362–1379.

[54] Devin Koepl and Jonathan Hurst. “Impulse control for planar spring-mass running”. In: *Journal of Intelligent & Robotic Systems* 74 (2014), pp. 589–603.

[55] P.V. Kokotovic. “The joy of feedback: nonlinear and adaptive”. In: *IEEE Control Systems Magazine* 12.3 (1992), pp. 7–17. DOI: 10.1109/37.165507.

[56] Scott Kuindersma et al. “Optimization-based locomotion planning, estimation, and control design for the ATLAS humanoid robot”. In: *Autonomous Robots* 40.3 (2016), pp. 429–455.

[57] Maxime Lecointe, Caroline Ponzoni Carvalho Chanel, and Francois Defay. “Backstepping control law application to path tracking with an indoor quadrotor”. In: (2015).

[58] Joonho Lee et al. “Learning quadrupedal locomotion over challenging terrain”. In: *Science robotics* 5.47 (2020), eabc5986.

[59] Chen Li, Tingnan Zhang, and Daniel I Goldman. “A terradynamics of legged locomotion on granular media”. In: *Science* 339.6126 (2013), pp. 1408–1412.

[60] Robert B McGhee and Andrew A Frank. “On the stability properties of quadruped creeping gaits”. In: *Mathematical Biosciences* 3 (1968), pp. 331–351.

[61] Daniel Mellinger and Vijay Kumar. “Minimum snap trajectory generation and control for quadrotors”. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 2520–2525.

[62] Aykut Özgun Önol, Philip Long, and Taşkın Padır. “Contact-implicit trajectory optimization based on a variable smooth contact model and successive convexification”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2447–2453.

[63] Nicholas Paine et al. “Actuator Control for the NASA-JSC Valkyrie Humanoid Robot: A Decoupled Dynamics Approach for Torque Control of Series Elastic Robots”. In: *Journal of Field Robotics* 32.3 (2015), pp. 378–396.

[64] Ramiro Ibarra Pérez et al. “Attitude control of a quadcopter using adaptive control technique”. In: *Adaptive Robust Control Systems* (2017).

[65] Giacomo Picardi, Cecilia Laschi, and Marcello Calisti. “Model-based open loop control of a multigait legged underwater robot”. In: *Mechatronics* 55 (2018), pp. 162–170.

[66] Giulia Piovan and Katie Byl. “Approximation and control of the slip model dynamics via partial feedback linearization and two-element leg actuation strategy”. In: *IEEE Transactions on Robotics* 32.2 (2016), pp. 399–412.

[67] Giulia Piovan and Katie Byl. "Reachability-based control for the active slip model". In: *The International Journal of Robotics Research* 34.3 (2015), pp. 270–287.

[68] Michael Posa, Cecilia Cantu, and Russ Tedrake. "A direct method for trajectory optimization of rigid bodies through contact". In: *The International Journal of Robotics Research* 33.1 (2014), pp. 69–81.

[69] Ioannis Poulakakis and Jessy W Grizzle. "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper". In: *IEEE Transactions on Automatic Control* 54.8 (2009), pp. 1779–1793.

[70] Matthew J Powell, Eric A Cousineau, and Aaron D Ames. "Model predictive control of underactuated bipedal robotic walking". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 5121–5126.

[71] James A. Preiss et al. "Crazyswarm: A large nano-quadcopter swarm". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 3299–3304. DOI: 10.1109/ICRA.2017.7989376.

[72] Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

[73] Hadi Razmi and Sima Afshinfar. "Neural network-based adaptive sliding mode control design for position and attitude control of a quadrotor UAV". In: *Aerospace Science and technology* 91 (2019), pp. 12–27.

[74] Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. "Unsupervised Contact Learning for Humanoid Estimation and Control". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 411–417.

[75] Nikita Rudin et al. "Advanced Skills by Learning Locomotion and Local Navigation End-to-End". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 2497–2503. DOI: 10.1109/IROS47612.2022.9981198.

[76] Ajay Sathya et al. "Embedded nonlinear model predictive control for obstacle avoidance using PANOC". In: *2018 European control conference (ECC)*. IEEE. 2018, pp. 1523–1528.

[77] Shaun Sawyer. "Gain-scheduled control of a quadcopter UAV". MA thesis. University of Waterloo, 2015.

[78] William John Schwind. *Spring loaded inverted pendulum running: A plant model*. University of Michigan, 1998.

[79] Andre Seyfarth et al. "A movement criterion for running". In: *Journal of biomechanics* 35.5 (2002), pp. 649–655.

[80] Junjie Shen and Dennis Hong. "Convex model predictive control of single rigid body model on so (3) for versatile dynamic legged motions". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 6586–6592.

[81] Maitray Shrivastava, Ashish Dutta, and Anupam Saxena. "Trajectory generation using GA for an 8 DOF biped robot with deformation at the sole of the foot". In: *Journal of Intelligent and Robotic Systems* 49.1 (2007), pp. 67–84.

[82] Koushil Sreenath et al. "Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on MABEL". In: *The International Journal of Robotics Research* 32.3 (2013), pp. 324–345.

[83] Thomas J Stastny, Adyasha Dash, and Roland Siegwart. "Nonlinear MPC for fixed-wing UAV trajectory tracking: Implementation and flight experiments". In: *AIAA guidance, navigation, and control conference*. 2017, p. 1512.

[84] Bartolomeo Stellato et al. "OSQP: An operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672.

[85] Oskar Von Stryk. "Numerical solution of optimal control problems by direct collocation". In: *Optimal control*. Springer, 1993, pp. 129–143.

[86] Michael Szmuk et al. "Convexification and real-time on-board optimization for agile quadrotor maneuvering and obstacle avoidance". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4862–4868.

[87] Yuval Tassa and Emo Todorov. "Stochastic complementarity for local control of discontinuous dynamics". In: *Robotics: Science and Systems VI* (2010).

[88] Johnathan Van Why et al. "Running into a trap: numerical design of task-optimal preflex behaviors for delayed disturbance responses". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 2537–2542.

[89] Miomir Vukobratovic and Davor Juricic. "Contribution to the synthesis of biped gait". In: *IEEE Transactions on Biomedical Engineering* 1 (1969), pp. 1–6.

[90] Ke Wang, Hengyi Fei, and Petar Kormushev. "Fast Online Optimization for Terrain-Blind Bipedal Robot Walking with a Decoupled Actuated SLIP Model". In: (2021).

[91] E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. "Hybrid zero dynamics of planar biped walkers". In: *IEEE Transactions on Automatic Control* 48.1 (2003), pp. 42–56. DOI: 10.1109/TAC.2002.806653.

[92] Jason White, Dylan Swart, and Christian Hubicki. "Force-based Control of Bipedal Balancing on Dynamic Terrain with the" Tallahassee Cassie" Robotic Platform". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6618–6624.

[93] Jason White et al. "Avoiding Dynamic Obstacles with Real-time Motion Planning using Quadratic Programming for Varied Locomotion Modes". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 13626–13633. DOI: 10.1109/IROS47612.2022.9981268.

[94] Jason White et al. "Avoiding Dynamic Obstacles with Real-time Motion Planning using Quadratic Programming for Varied Locomotion Modes". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 13626–13633.

[95] Jason White et al. "Online Gait Optimization for Running in Resistive Media". In: *2022 7th International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE. 2022, pp. 282–286.

[96] Georg Wiedebach et al. "Walking on partial footholds including line contacts with the humanoid robot atlas". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2016, pp. 1312–1319.

[97] Albert Wu and Hartmut Geyer. "The 3-D spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments". In: *IEEE Transactions on Robotics* 29.5 (2013), pp. 1114–1124.

[98] Weitao Xi, Yevgeniy Yesilevskiy, and C David Remy. "Selecting gaits for economical locomotion of legged robots". In: *The International Journal of Robotics Research* 35.9 (2016), pp. 1140–1154.

[99] Zhaoming Xie et al. "Feedback Control For Cassie With Deep Reinforcement Learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1241–1246. DOI: 10.1109/IROS.2018.8593722.

[100] Zhaoming Xie et al. "Iterative reinforcement learning based design of dynamic locomotion skills for cassie". In: *arXiv preprint arXiv:1903.09537* (2019).

[101] Xiaobin Xiong and Aaron D Ames. "Coupling reduced order models via feedback control for 3d underactuated bipedal robotic walking". In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 1–9.

[102] Xiaobin Xiong, Aaron D Ames, and Daniel I Goldman. "A stability region criterion for flat-footed bipedal walking on deformable granular terrain". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4552–4559.

[103] Andrew Zulu and Samuel John. "A review of control algorithms for autonomous quadrotors". In: *arXiv preprint arXiv:1602.02622* (2016).

BIOGRAPHICAL SKETCH

Jason White was born and grew up in Baltimore, Maryland. After completing his schoolwork at Calvert Hall College High School in 2008, Jason entered Virginia Polytechnic Institute and State University in Blacksburg, Virginia. In May 2012, he received a Bachelor of Science with a major in Civil Engineering and a minor in Green Engineering. He then worked in Maryland as a residential and commercial site design engineer until moving to San Francisco, California in November 2014. There he worked as a structural field engineer and construction manager until ultimately deciding to change careers and pursue a degree focused on robotics. In August of 2018, he enrolled as a Mechanical Engineering Ph.D. student at the FAMU/FSU College of Engineering under Christian Hubicki. His research primarily focuses on gait generation for legged locomotion but also includes developing a method of dynamic obstacle avoidance across locomotion modes, quadrupedal design prototyping, and control of an articulated wheeled vehicle. Furthermore, he has utilized his controllers on multiple hardware platforms including the Cassie biped from Agility Robotics, Crazyflie drones from Bitcraze, a Minitaur hopper from Ghost Robotics, and the in-house developed quadruped ET-Quad. Jason is currently in his 6th year of study at Florida State and will graduate in December 2023.