Real-Time Planning and Control of Robots in Dynamic Environments

by

Jason White

A Prospectus Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Graduate Supervisory Committee:

Christian Hubicki, Chair
Jonathan Clark
Brandon Krick
Rodney Roberts

FLORIDA STATE UNIVERSITY

July 20th 2023

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 colleges. Thank you all for taking the time to teach me and answer any questions when I first arrived with little to no knowledge in the field of robotics. Thank you for spending countless hours working on projects, or staying late to meet deadlines, much of the work here would not exist without your help.

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

TABLE OF CONTENTS

Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Performance of Robots

Robots have long held the promise to mimic or exceed the reflexes, agility, and abilities of animals. Although some robots perform repetitive tasks far more accurate and faster than their human counterpart (e.g. manufacturing pick-and-place task), unexpected circumstances can quickly lead to failed task or damaged hardware. This limitation has constrained the ability of robots to perform the broader scope of tasks our society relies on. Of the robots found in homes, robotic vacuum cleaners are arguable the most common. They can dynamically map and plan surrounding environments but still struggle with full autonomy, constantly getting stuck or damaged. Commercial drones use image recognition to follow an individual, while mapping and avoiding stationary obstacles. However, it likely struggles when it's environment rapidly changes. Boston Dynamic's Atlas demonstrates very impressive athleticism from dancing, to back-flips, and basic manipulation of large objects. Unfortunately, we don't know how what the limits of this platform are, for instance how would it reacts to a terrain that is not ridged or collapses beneath its feet? This thesis will seek to address the gap that still exist when encountering dynamic environments such as unstable terrain, moving obstacles, and legged locomotion through viscous media.

1.2 Dynamic Legged Robots

Early forms of legged robots used more static control strategies to remain stable. For bipedal robots a classic control approach called zero moment point [1] keeps the

center of pressure saftley within it's polygon of foot support to prevent tipping, but limits the range of dynamic behaviors available to the robot. Quadrupeds use similar methods for "creeping" gaits [2] which remain stable throughout the gait by always maintaining three points of contact.

Later approaches begin to treat walking and running as a dynamic task. Such gaits can 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. Among the earliest examples of this is the Raibert hopper from the 1980's [3] which hops and performs front flips despite having uncontrolled ballistic trajectories while airborne.

Modern bipedal robots are designed to withstand larger impulse/impact forces, generally synonymous with periodic stability. Controlling these more dynamic gaits becomes challenging since impact forces are discontinuous, and footholds may not be rigid. Recent research shows addressing these complications is possible by having hardware platforms perform complicated maneuvers such as backflips for both bipeds and quadrupeds [4]. 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 can even balance on two legs [5].

These feats are generally accomplished using model-based controllers which leverage knowledge of the robots system's dynamics. For control at a given time step operational space control, force control, and hybrid zero dynamics are all ways to develop desired motor torques. For task-based planning, optimizations are a common choice and are used for whole body control, direct collocation, model predictive control, etc. There has also been a lot of success with reinforcement learning in legged robotics. Whether starting with a reference gait or developing one from scratch, RL training in both simulation and hardware have created controllers which can train in

minutes, and tackle a broad spectrum of terrains and obstacles. This thesis leverages novel combinations of these approaches to enable legged locomotion in more dynamic environments.

1.3 Dynamic Aerial Robots

Since the early 2000's UAVs have vastly grown in popularity with quad-copters further rising to prominence in the 2010's. These underactuated systems can 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 [6], [7]. Today everything from linear quadratic regulator's (LQR) [8] to the more recent differential flatness [9] are techniques to optimize control of UAVs. Other controllers such as gain scheduling [10] which use variable gains with linear dynamics, and adaptive control [11] which automatically adjust control parameters in real time, embrace the power of reference tables for control. These are only a few of the options available to controls engineers, neural networks [12], fuzzy logic [13], and many more can be put onto a UAV depending on its use. What these controllers have in common, is many of them are combined with higher level planning which include 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 which avoids dynamic obstacles.

1.4 Challenges of Dynamic Control

Dynamic environments rapidly change, at any moment a controller may need to adjust to avoid an obstacle, react to an unexpected terrain, or adapt a 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 model a ground environment we would need all slopes, terrain, ground stiffness, ground dampening factors, media resistance properties, etc. Developing this could take years, instead we would rather design a controller which 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 plan for them are more robust in unstructured environments.

To accomplish this we focus on minimizing the complexity of the models, or so called reduced order models. Avoiding the use of detailed models allows us to devise control computations which solve quickly 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 optimizations. The time to develop desired motor toques drops to 2kHz using force control, planning through viscous media takes 1 sec. rather than 20 mins., and developing trajectories around moving obstacles happens more than 100 times per second.

1.5 Outline

This prospectus is divided into three parts. Part I considers a method of reaction to dynamic terrain for a high degree of freedom biped, and introduces hardware results validating the proposed method. Part II again considers a variation of dynamic terrain, motion of a single legged hopper through a viscous media. Part III discusses a method of avoiding dynamic adversarial obstacles.

1.6 Contribution

The scope of this thesis touches on several research topics from optimal control to force control and legged locomotion to UAV's, all of which are validated with hardware experiments. Each section details specific gaps in the field and how the proposed work contributes to filling that gap. Furthermore the chapters presented here are reprinted from my published work.

Chapter 2

BIPEDAL FORCE CONTROL

2.1 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.

The conceptual pillar of bipedal balancing is the Zero-Moment Point (ZMP) [14] and its popular embodiment in Preview Control [15]. 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 [16]. 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 [17] - a form of dynamic terrain. Some DRC methods adjusted for unintended robot compliance by modeling it as a quasistatic error [18].

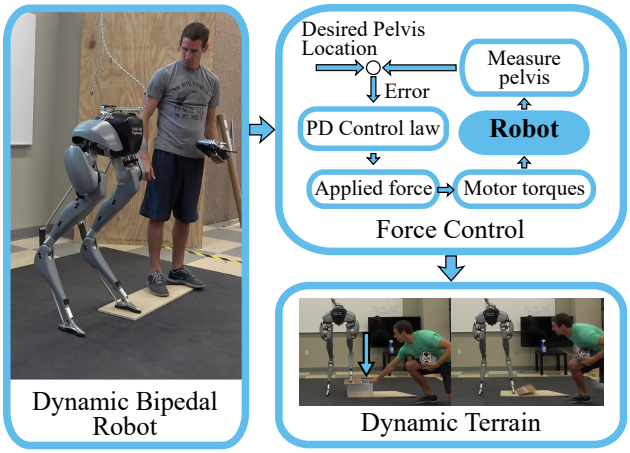

Figure 2.1: We implement a force-based controller for balancing the bipedal robot, "Tallahassee Cassie" on dynamic terrain, such as soft terrain or a sudden loss of foot contact.

To extend the capability of humanoids, control methods for these fully-actuated machines have been adapted in various ways to explicitly accommodate for deformable contact. 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 [19]. 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 [20]. 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 [21], and some work even uses machine learning to estimate the properties of ground-contact in-situ [22], [23], [24].

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 [25], uses an estimate of the robot's foot roll and torque control to correct its balance as the ground deforms.

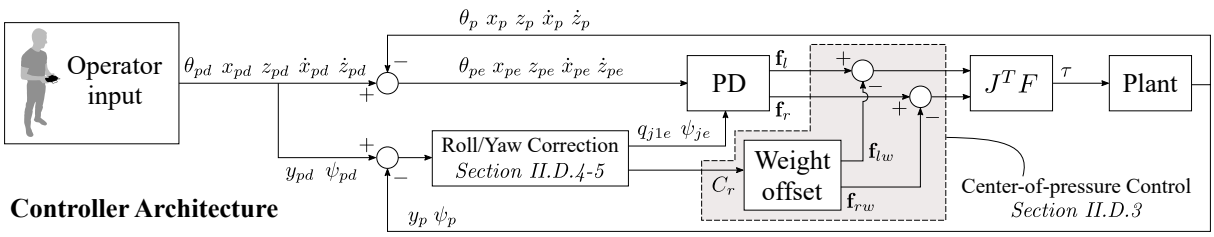

Figure 2.2: Overview of control structure detailed in Section II.

Taking a more-general force/torque control approach, NASA's Valkyrie is built with series compliance to enable torque control [26], and DLR's TORO robot [27] was torque-controlled to balance on soft gymnasium mats [28], 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 [29] and the use of stepping to balance in 3D [30]. They do so using a whole-body variant of operational space control (OSC) [31] 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 [32] and slippery surfaces [33].

Finally, multiple control methods 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 [34] and a combined hybrid-nonlinear/machine-learned policy [35]. Methods for controlling Cassie robots to walk include deep reinforcement policy learning [36] and feedback control based

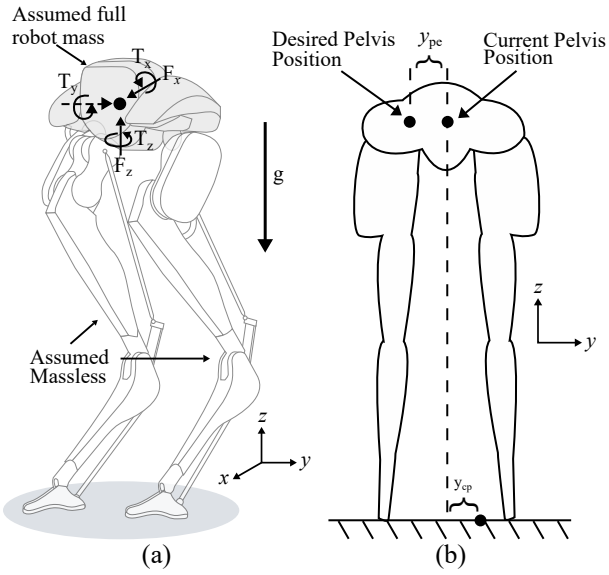

Figure 2.3: (a) Cassie is controlled by commanding the listed forces and torques on the pelvis. The pelvis is assumed to be the only component with mass. (b) Cassie can regulate its y position of its pelvis by controlling the desired center of pressure.

on reduced-order models [37]. Using control outputs that are kinematically defined, "Cassie Blue" was able to stand on uneven surfaces and walk on a variety of outdoor terrains [38]. 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. *We present the ability to maintain balance in the face of soft and rapidly-moving surfaces (i.e. dynamic terrain) including foam surfaces and sudden drops while using few modeling assumptions.*

2.2 Methods

2.2.1 Conventions

All quantities in the body coordinate system have a left subscript **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.

2.2.2 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 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 multiplication of the Jacobian transpose. We denote the current angles and positions in the world frame as

$$\mathbf{p}_i = \begin{bmatrix} \phi_i & \theta_i & \psi_i & x_i & y_i & z_i \end{bmatrix}^T \tag{2.1}$$

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

$$\mathbf{p}_{id} = \begin{bmatrix} \phi_{id} & \theta_{id} & \psi_{id} & x_{id} & y_{id} & z_{id} \end{bmatrix}^T \tag{2.2}$$

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, where as 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.

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

Here, $[q_{i1} \ldots q_{i5}]$ represents motor angles, and $[{}_Bp_{i1} \ldots {}_Bp_{i6}]^T$ 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.

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

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.

2.2.3 "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.

2.2.4 Controller Implementation

We make the following key assumptions to develop the controller. First, the low friction cycloidal drives allow us to assume 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 is able to resist external forces up to a limit dependent upon ground slope, damping coefficient, and foot contact friction. The height of the pelvis is defined by the foot

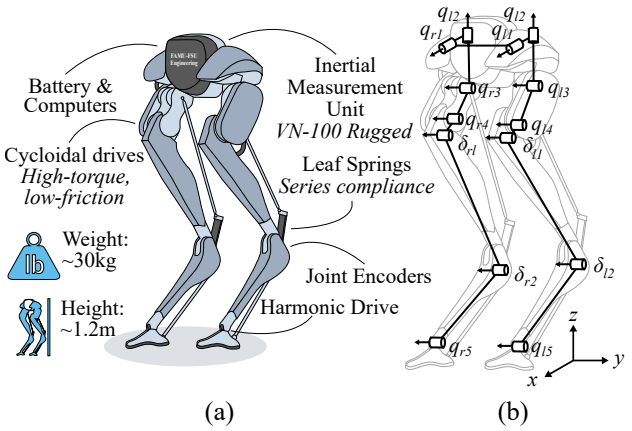

Figure 2.4: (a) Overview of Cassie's hardware and basic features. (b) Kinematic diagram of Cassie showing motor positions and spring deflections.

with the largest z distance as can been 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 error between the current and desired state. The system is treated similar 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)

$$F_{rz} = W C_r \tag{2.11}$$

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

$$\mathbf{f}_{jw} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & F_{jz} \end{bmatrix}^T \tag{2.13}$$

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 feet 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

In order 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

$$\mathbf{p}_{je} = \begin{bmatrix} \mathbf{q}_{j1e} & \theta_{pe} & \boldsymbol{\psi}_{je} & x_{pe} & 0 & z_{pe} \end{bmatrix}^T \tag{2.19}$$

and

$$\mathbf{v}_{je} = \begin{bmatrix} 0 & 0 & 0 & \dot{x}_{pe} & 0 & \dot{z}_{pe} \end{bmatrix}^T. \tag{2.20}$$

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.

2.2.5 Multibody Simulation

To test how the controller performs in simulation, we 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.

2.3 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. In order 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.

2.4 Results

Each of the four perturbation scenarios are shown visually in Fig. 2.6 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 a soft foam and pushed, and (4) when a 17cm-tall box is pulled from under its foot. In each case, the controller accommo-

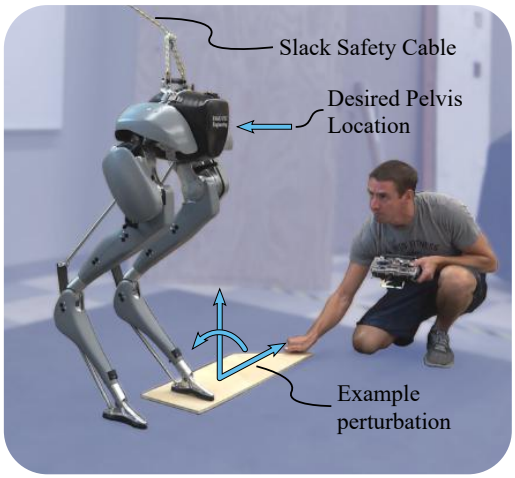

Figure 2.5: Experimental setup for testing the balancing controller on Tallahassee Cassie for each of four balance perturbations.

dates the disturbance, and settles back near the desired pelvis location. These tests experimentally demonstrate the effectiveness of the simple force controller in spite of highly simplified assumptions.

2.5 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 a pulling out a supporting box, the force controller quickly stomped the leg 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 drives 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 in spite of the vastly simplifying

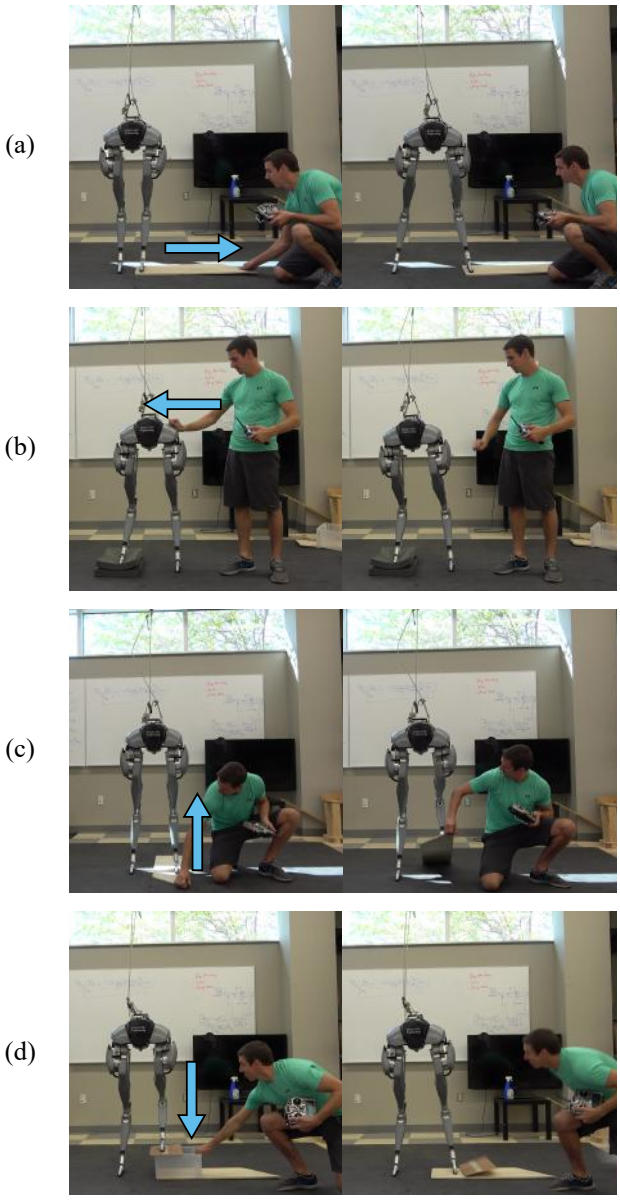

Figure 2.6: Examples of different perturbations applied on Tallahassee Cassie. (a) "Leg Spread": Cassie's foot is moved by sliding a board. (b) "Soft Foam and Push": One of Cassie's feet is supported by soft foam and the pelvis is pushed. (c) "Lift and Lower": Cassie's foot is raised and lowered using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

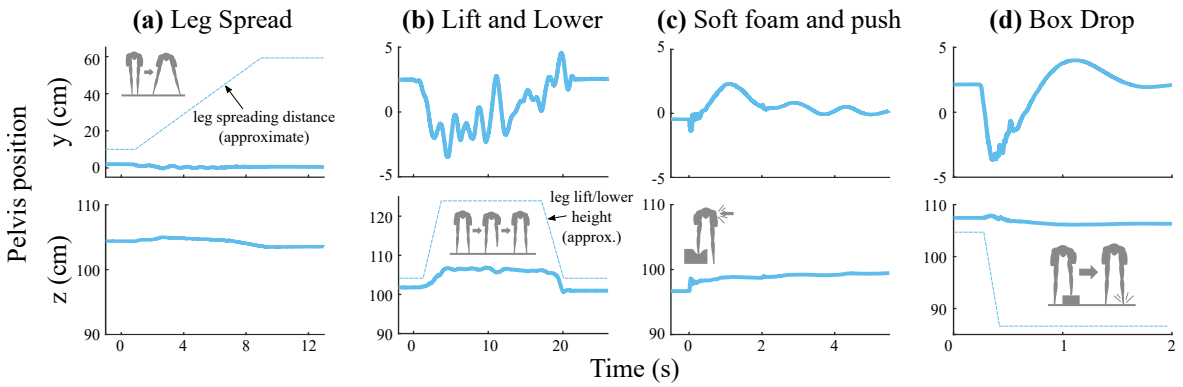

Figure 2.7: Experimental data from each of four perturbation tests. (a) Cassie's foot is pulled away while maintaining a stable height. (b) Cassie's foot is moved up and down and Cassie returns to its initial position. (c) Cassie is pushed on soft foam and returns to a stable position. (d) Cassie's foot falls rapidly as a box is pulled out from beneath its foot, and Cassie recovers to 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.

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.

Acknowledgment

This work was supported by the Mechanical Engineering Department in the FAMU-FSU College of Engineering, the Florida State University Council on Research and Creativity, and the collaborative participation in the Robotics Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD 19-01-20012.

Chapter 3

SLIP CONTROL IN RESISTIVE MEDIA

3.1 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 paper extends state-of-the-art running control methods to include a computationally tractable fluid dynamic model to generate efficient locomotion through resistive media (Fig. 3.1A). 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 [3], but modern methods use model-based optimization to synthesize behaviors. While system models can be detailed with multibody dynamics [39, 40], reduced-order models can synthesize a variety of behaviors using simpler and faster computations [32, 41]. 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 [42]. The SLIP model is credited with inspiring successful leg control schemes [43, 44], but it is energetically passive and thus needs 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

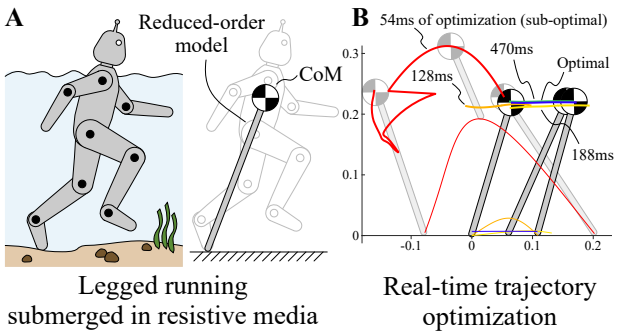

Figure 3.1: This paper presents rapid optimization of legged locomotion in resistive media (e.g. fluids) A. An illustration of how a multi-body legged robot is abstracted to a reduced-order FF-SLIP model. B. The Cartesian CoM and foot trajectories of a gait during optimization, achieving near-optimal behavior in 470ms.

in rough terrain [45, 46].

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 [47] (or Underwater SLIP), took a lumped modeling approach for synthesizing SLIP model gaits in a resistive media generating underwater "punting" gaits [48]. While the U-SLIP model ignored resistive forces on the leg, a subsequent approach modeled leg drag and generated gaits through different fluid depths [49]. 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 [50]. This method of trajectory generation accommodates a more complex dynamic model

but requires some 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 paper 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 [50] 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 [51] suggesting that the optimization is successfully minimizing energy waste. As a final proof of concept, we implement an example optimized gait (Fig. 3.1B) 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. 3.5C).

3.2 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.

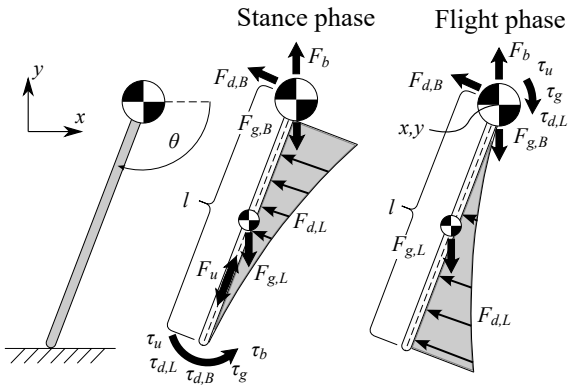

Figure 3.2: Diagram of the presented reduced-order legged model in stance (foot contact) and flight (no foot contact) phases.

3.2.1 Conventions

Sub-scripted capital letters refer to forces applied to either the body (F_B) or the leg (F_L). Lower case 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

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

3.2.2 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 actuated Spring-loaded Inverted Pendulum (SLIP) model. While the approximation makes several simplifying assumptions similar to [50], 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. 3.2). 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:

$$\begin{aligned} F_g &= -\underbrace{m_B g}_{F_{g,B}} - \underbrace{m_L g}_{F_{g,L}} \\ \tau_g &= -gl\left(m_B + \frac{m_L}{2}\right)\cos(\theta) \end{aligned} \tag{3.1}$$

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,

$$\begin{aligned} F_b &= \rho V_B g \\ \tau_b &= \rho V_B g l \cos(\theta). \end{aligned} \tag{3.2}$$

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:

$$
\begin{aligned}
F_{d,B} &= -\frac{\rho C_B A_B \dot{l}|\dot{l}|}{2} \\
\tau_{d,L} &= -\frac{\rho C_L W_L \dot{\theta}|\dot{\theta}| l^4}{8} \\
\tau_{d,B} &= -\frac{\rho C_B A_B \dot{\theta}|\dot{\theta}| l^3}{2}
\end{aligned}
\tag{3.3}
$$

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

$$
\begin{aligned}
(m_B + m_L)\ddot{l} &= F_u + F_{d,B} - F_g \sin\theta - F_b \sin\theta \\
(m_B + \frac{m_L}{3}) l^2 \ddot{\theta} &= \tau_u + \tau_{d,B} + \tau_{d,L} - \tau_g - \tau_b.
\end{aligned}
\tag{3.4}
$$

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

$$
\tau_g = \frac{m_L g l \cos(\theta)}{2}. \tag{3.5}
$$

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 true. We instead find the velocity of the body CoM, v_L, as per:

$$v_L = \dot{y}\cos\theta - \dot{x}\sin\theta, \tag{3.6}$$

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:

$$\begin{aligned} F_{d,B_x} &= -\frac{\rho C_B A_B \dot{x}|\dot{x}|}{2}, \quad F_{d,B_y} = -\frac{\rho C_B A_B \dot{y}|\dot{y}|}{2} \\ F_{d,L} &= -\frac{\rho C_L W_L l(\dot{\theta}|\dot{\theta}|l^2 + 3\dot{\theta}v_L l + 3v_L|v_L|)}{6} \\ \tau_{d,L} &= -\frac{\rho C_L W_L l^2(3\dot{\theta}|\dot{\theta}|l^2 + 8\dot{\theta}v_L l + 6v_L|v_L|)}{24}. \end{aligned} \tag{3.7}$$

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

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

3.2.3 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. 3.5B). 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 [50], 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

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 3.1: Fluid-field SLIP model (FF-SLIP) parameters

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 $\left(\ddot{l}\right)$ is defined during flight via a forced second-order model using an actuation force (F_u) and an added damping term (c) per:

$$m_L\ddot{l} = F_u - c\dot{l} - m_L g. \tag{3.8}$$

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 $\epsilon \ll 1$.

3.2.4 Optimal Control Problem

The optimal control problem is cast as a nonlinear direct collocation trajectory optimization [52].

$$\begin{aligned} \min_{q,u,t} \quad & f(q,u,t) \\ \text{s.t.} \quad & c_1(q,u,t) = 0 \\ & c_2(q,u,t) \leq 0 \end{aligned} \tag{3.9}$$

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,

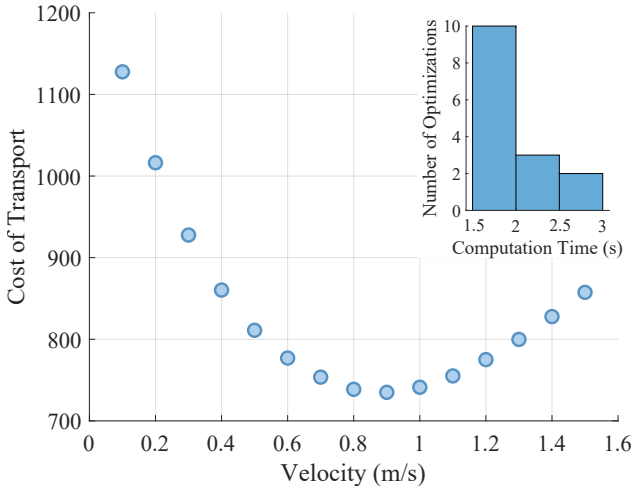

Figure 3.3: Cost of transport and solve times for gaits of varied speeds. The COT curve is smooth and U-shaped – a common feature of energy-minimizing locomotion that suggests the optimization is avoiding pseudominima.

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 in a similar manner, by constraining to the initial mode 1 states to equal the final mode 2 states,

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

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,

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

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:

$$\begin{aligned} 0.1 \leq x_{final} - x_{init} \qquad & \text{(Forward motion)} \\ 0 \leq F_l \qquad & \text{(Unilateral leg force).} \end{aligned}$$

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. 3.2) to a five-bar leg morphology (Fig. 3.4A) using a kinematic relationship. We use the law of cosines to find interior leg angles

$$\cos(\phi) = \frac{l_1^2 + l_2^2 - l^2}{2l_1 l_2}, \quad \cos(\psi) = \frac{l_2^2 + l^2 - l_1^2}{2l_2 l} \tag{3.10}$$

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 caused 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

$$\tau_{M1} = \frac{F_u l_1}{2\cos\phi\sin\psi} + \frac{\tau_u}{2}, \quad \tau_{M2} = \frac{F_u l_1}{2\cos\phi\sin\psi} - \frac{\tau_u}{2} \tag{3.11}$$

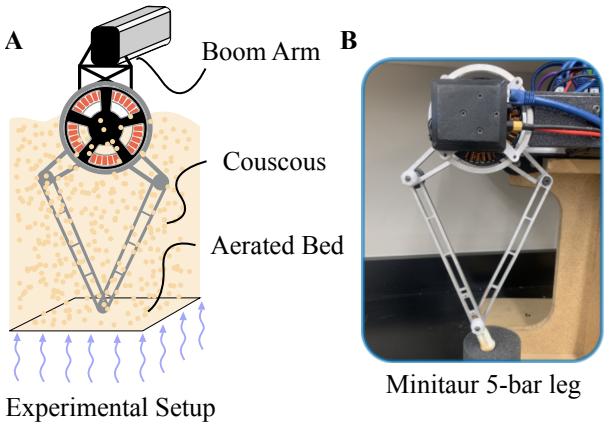

Figure 3.4: A. Schematic of a two-motor five-bar monopod robot planarized by rigid attachment to a 2-DOF boom arm. Fluid resistance is emulated by an aerated bed of dry granular media (couscous). B. A five-bar "Minitaur" robot leg.

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

$$|\tau_{M1}|,\ |\tau_{M2}| \leq 5.91. \tag{3.12}$$

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:

$$\text{CoT} = \left(\int_0^t |\tau_{M1}| + |\tau_{M2}| dt \right) / (x_{final} - x_{init}). \tag{3.13}$$

We then export the NLP's costs, constraints, and their analytical derivatives using the MATLAB framework COALESCE [53]. We then solve the optimization using the open-source large-scale NLP solver, IPOPT [54].

3.3 Results

3.3.1 Numerical Results

Using the parameters seen in Table 3.1, we generate trajectories which 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.

3.3.2 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. 3.5A), similar to [55]. 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 in order 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

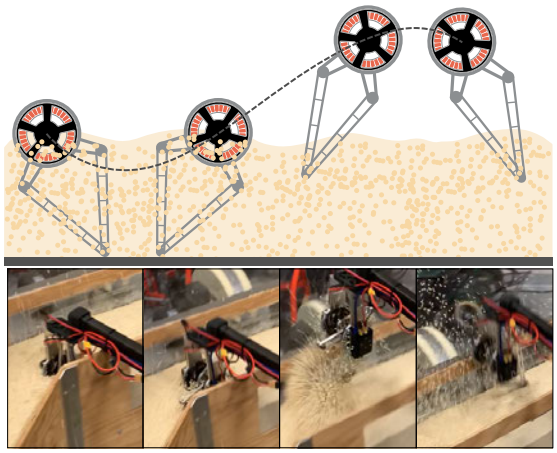

Figure 3.5: Demonstration of fast-optimized monopod hopping through aerated couscous–a surrogate fluid-like substrate.

simulation, as depicted in tiles in Fig. 3.5.

3.4 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 (*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*.

Acknowledgment

Authors JW, MA, JC, and CH are with the FAMU-FSU College of Engineering in Tallahassee, FL USA. Author JP is with the U.S. Army CCDC Army Research Laboratory in Adelphi, MD USA.

Research was sponsored by the DEVCOM Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-16-2-0008. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 4

DRONE AVOIDANCE OF DYNAMIC OBSTACLES

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 to 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" [56]. 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 disallowing 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 [57] which have since been unified across agent plant dynamics [58]. However, these guarantees are less reliable under the practical actuation limitations of robots. Other methods explicitly use mutual communication to update collision-free trajectories [59]. Further, safety-focused control methods like barrier functions [60] 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 Pre-

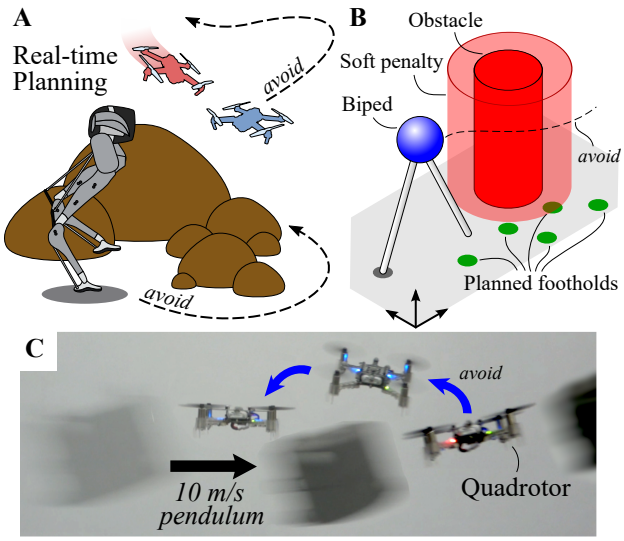

Figure 4.1: **A.** Robots must quickly plan their motions to avoid obstacles and adversaries in real-world environments. **B.** Our optimization methods use linear constraints and costs to achieve reliable real-time motion planning that spans locomotion modes from flying to walking. **C.** Demonstration of a quadrotor evading a 10m/s swinging pendulum. Note: drone positions were adjusted for image clarity, see supplementary video for unadjusted footage.

dictive Control, or MPC) to flexibly generate control on-the-fly for quadrotors [61], fixed-wing aircraft [62], quadrupeds [32], bipeds [63]. Further, these methods can be more robust to dynamical assumptions about obstacles. MPC has the additional benefit of minimizing a cost function while respecting constraints, (*e.g.* physical limits, and obstacles [64]). 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 [65]. 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 [66] which solve at 8-30Hz, and fast nonlinear solvers like SQP [67] and embedded solvers like

PANOC [68] 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 [69]. We leverage a half-space convex relaxation of the obstacle-free space [70] 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 paper 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.

The overall contributions of this paper are:

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 on quadrotor hardware in dynamic avoidance tasks.

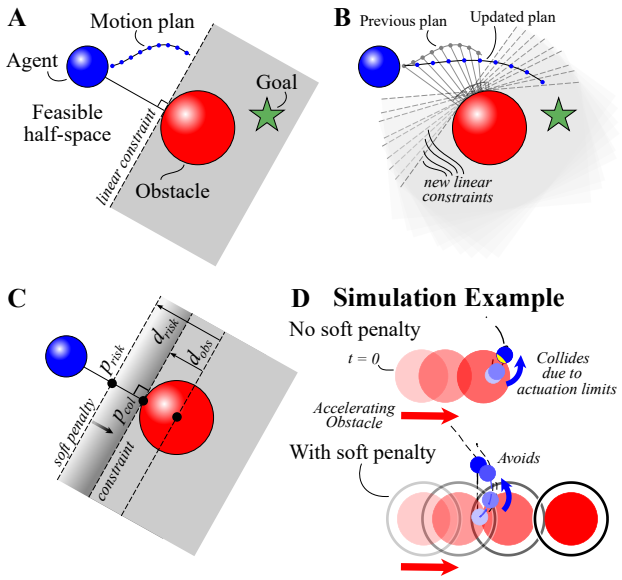

Figure 4.2: **A.** A half-space relaxation slices the feasible space in half with the plane tangent to the closest obstacle point. **B.** Half-spaces are recut using the previous planned trajectory when computing the closest obstacle point, iteratively improving the half-space approximation. **C.** We 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.

4.1 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.

4.1.1 Conventions

Scalar values are italicized (J); vectors are bold, upright, lowercase ($\mathbf{q}$); and matrices are bold, upright, uppercase ($\mathbf{A}$). Variables ($\mathbf{p}$, $\mathbf{v}$, & $\mathbf{u}$) are positions, velocities, and control inputs respectively. A right subscript references specific nodes within the trajectory; ($\mathbf{q}_k$) for node k, ($\mathbf{q}_{k+1}$) for node $k+1$, and ($\mathbf{q}_N$) for the final node. Right subscripts with a comma ($\mathbf{q}_{prev,k+1}$) indicate a description of the variable, followed by the node number. $\mathbf{W}_\star$ denote positive semi-definite symmetric weighting matrices for cost category, $\star$. We define one system using a double integrator model in three Cartesian dimensions ($j = 3$), for (x, y, z), and a second system, the LIPM, in two dimensions ($j = 2$) for (x, y). We define velocities, control inputs, and states respectively as

$$\mathbf{v}_k \in \mathbb{R}^j \quad \mathbf{u}_k \in \mathbb{R}^j \quad \mathbf{q}_k = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \end{bmatrix}. \tag{4.1}$$

4.1.2 Assumptions

We assume the initial positions (i.e. the measured positions) ($\mathbf{p}_{meas}$) and velocities ($\mathbf{v}_{meas}$) of both the agent ($\mathbf{q}$) and obstacle ($\mathbf{q}_{obs}$) 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. 4.1.4 uses only the obstacle's initial states ($\mathbf{q}_{obs,meas}$) with a simple single integrator assumption of the dynamics. Specifically, the obstacle's future motion is estimated using only its current states $\mathbf{q}_{obs,meas}$ integrated forward in time, ΔT, per:

$$\mathbf{p}_{obs,k} = \mathbf{p}_{obs,meas} + \mathbf{v}_{obs,meas}\Delta T(k-1). \tag{4.2}$$

Despite these simplifying model assumptions, we will demonstrate its effective avoid-

ance behaviors even when implemented in crowded multiagent scenarios (Sec. 4.2.4). 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.

4.1.3 Model Predictive Control

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 trajectories (**p,v**) and associated control inputs (**u**) while avoiding obstacles–facilitated in part by including a slack variable (ρ).

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 ($\mathbf{q}_{meas}$) to equal the initial trajectory states ($\mathbf{q}_1$). The optimization uses a multi-part quadratic cost, and linear constraints per:

$$\min_{\mathbf{q}_k, \mathbf{u}_k, \rho_k} \quad \underbrace{J(\mathbf{q}, \mathbf{u}, \mathbf{s})}_{\text{Task}} + \sum_{k=1}^{N} \underbrace{\|\rho_k\|^2_{\mathbf{W}_{\text{avoid}}}}_{\text{Avoidance}}$$

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

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).

4.1.4 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. 4.2A). 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

$$\mathbf{d}_k = \mathbf{p}_{prev,k} - \mathbf{p}_{obs,k} \tag{4.3}$$

by using the previously generated motion plans ($\mathbf{p}_{prev,k}$) as seen in (Fig. 4.2B). The obstacle collision point ($\mathbf{p}_{col,k}$) in Cartesian space is estimated by simply defining the obstacle depth ($\mathbf{d}_{obs,k}$), and converting the distance ($\mathbf{d}_k$) into a unit vector ($\hat{\mathbf{d}}_k$), as per

$$\mathbf{p}_{col,k} = \mathbf{p}_{obs,k} + \mathbf{d}_{obs,k}\ \hat{\mathbf{d}}_k, \quad \hat{\mathbf{d}}_k = \frac{\mathbf{d}_k}{||\mathbf{d}_k||_2}. \tag{4.4}$$

The half-space constraint can then be written as:

$$\mathbf{d}_k \cdot (\mathbf{p}_{col,k} - \mathbf{p}_k) \leq 0. \tag{4.5}$$

This method alone only produces a rough estimate of the obstacle future states, and thus commonly violates constraints and leads to infeasible optimizations. To

resolve this, we define an additional "at risk of collision" Cartesian point ($\mathbf{p}_{risk,k}$) along the distance vector, closer to the agent, as seen in (Fig. 4.2). We develop a second planar cut, parallel to the first, but include an additional design variable in the equation (Fig. 4.2C). This design variable (ρ_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 peicewise formulation ensures there is no penalty outside of ($\mathbf{p}_{risk,k}$), but escalates quickly after crossing the ($\mathbf{p}_{risk,k}$) threshold. These constraints are written as such:

$$\begin{aligned} -\rho_k &\leq 0 \quad \text{(Slack No-penalty Region)} \\ \mathbf{d}_k \cdot (\mathbf{p}_{risk,k} - \mathbf{p}_k) - \rho_k &\leq 0 \qquad \text{(Slack Penalty Region)} \end{aligned} \tag{4.6}$$

where the slack variable (ρ) is included in the cost function to enforce the penalties associated with the agent encroaching on the obstacle's defined threshold,

$$J_{\text{avoid}}(\rho) = \sum_{k=1}^{N} ||\rho_k||^2_{\mathbf{W}_{\text{avoid}}}. \tag{4.7}$$

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.

4.1.5 Double Integrator Model Example

We introduce the state space model for a double integrator (Eq. 4.8) 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:

$$\dot{\mathbf{q}}_k = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{q}_k + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{u}_k. \tag{4.8}$$

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. 4.1.7.

4.1.6 Linear Inverted Pendulum Example

The Linear Inverted Pendulum Model (LIPM) has been widely used in control design for legged robots, particularly motion generation [71]. 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. (4.8A). The dynamics of the LIPM are

$$\dot{\mathbf{q}}_k = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{M} & \mathbf{0} \end{bmatrix} \mathbf{q}_k - \begin{bmatrix} \mathbf{0} \\ \mathbf{M} \end{bmatrix} \mathbf{u}_k; \quad \mathbf{M} = \begin{bmatrix} \frac{g}{z_0} & 0 \\ 0 & \frac{g}{z_0} \end{bmatrix} \tag{4.9}$$

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. 4.9) 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 [72, 73].

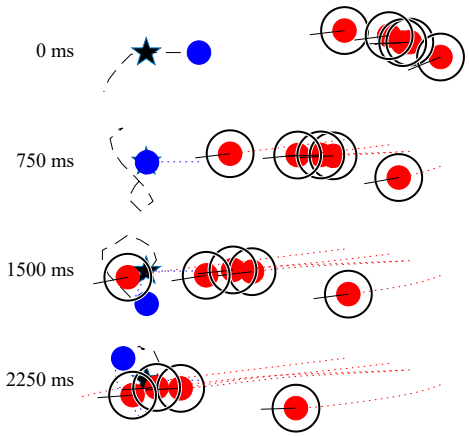

Figure 4.3: Simulated results of 2D uncoordinated waypoint tracking single agent (blue) with multi-adversary (red) avoidance. Despite the half-space constraints, the agent planned figure-eight looping maneuvers around adversaries. The four images show the tracking and avoidance trajectory through time (black) and the obstacle's traveled paths (red) over a 2.25 second period.

Reachability Constraints

A reachability constraint is included to ensure the planned footstep locations are physically achievable within the robot leg kinematics, and the reachability constraint can be defined in the following linear form:

$$h(\mathbf{q}_k, \mathbf{u}_k) = \mathbf{r}_{min} \leq (\mathbf{u}_k - \mathbf{q}_k) \leq \mathbf{r}_{max} \tag{4.10}$$

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

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 ($\mathbf{s}_{ref}$). This drives the

robot foot to the desired position from the current foot placement, and is formulated as such:

$$J_{\text{step}}(\mathbf{s}) = \sum_{k=1}^{N_s} ||\mathbf{s}_k - \mathbf{s}_{ref,k}||^2_{\mathbf{W}_{\text{step}}} \tag{4.11}$$

where N_s is the number of robot steps predicted, and the number of MPC prediction steps is found using $N = N_s \frac{T_{step}}{\Delta T}$. The sagittal ($s_{ref,x}$) and frontal ($s_{ref,y}$) desired (or reference) step lengths are computed per:

$$\mathbf{s}_{ref} = \begin{bmatrix} \dot{s}_{ref,x} & T_{step} \\ 2s_{foot,y} & (-1)^m \end{bmatrix} \tag{4.12}$$

using the x directional walking speed ($\dot{s}_{ref,x}$), stepping time (T_{step}), and the nominal y distance between the foot and robot CoM ($s_{foot,y}$). This is used to keep the feet away from each other and prevent self-collision. Term m indicates the ground contact foot with $m \in \{0, 1\}$ being the right and left foot respectively.

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

$$\mathbf{u}_{ref,x} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \dots \\ \dots \\ \mathbf{1} \end{bmatrix} p_{foot,x,1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots \\ \mathbf{1} & \mathbf{1} & \dots & \mathbf{1} \end{bmatrix} \begin{bmatrix} s_{x,1} \\ s_{x,2} \\ s_{x,3} \\ \dots \\ s_{x,N_s} \end{bmatrix} \tag{4.13}$$

where $\mathbf{0}$ and $\mathbf{1}$ are column vectors with length $N_r = \frac{T_{step}}{\Delta T}$. The x directional current foot placement is represented as ($p_{foot,x,1}$) and x directional predicted step length as (s_x). The frontal plane center of pressure reference ($\mathbf{u}_{ref,y}$) is found by substituting $p_{foot,y,1}$ and $s_{x,1:N_s}$ in (Eq. 4.13).

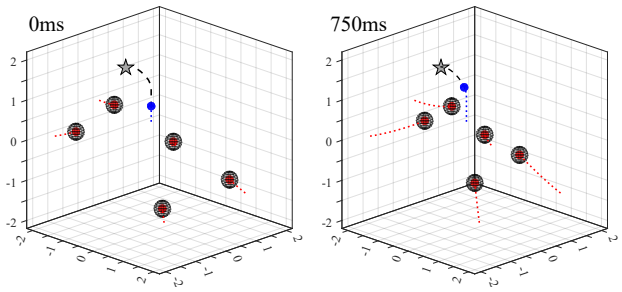

Figure 4.4: Simulated results of 3D uncoordinated waypoint tracking single agent (blue) with multi-adversary avoidance. Left and right images are sampled 0.75s apart.

4.1.7 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,

$$J_{\text{target}}(\mathbf{q}) = \sum_{k=1}^{N} \|\mathbf{q}_k - \mathbf{q}_{ref,k}\|^2_{\mathbf{W}_{\text{target}}}, \tag{4.14}$$

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 certain velocity).

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

$$J_{\text{effort}}(\mathbf{u}) = \sum_{k=1}^{N} \|\mathbf{u}_k - \mathbf{u}_{ref,k}\|^2_{\mathbf{W}_{\text{effort}}}. \tag{4.15}$$

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

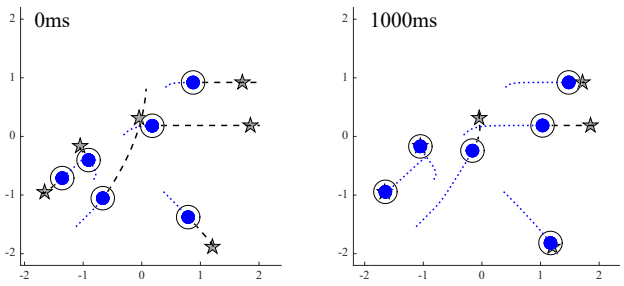

Figure 4.5: Simulated results of 2D uncoordinated multi-agent avoidance and waypoint tracking. Black dashed lines are motion plans. Left and right images sampled 1 second apart.

demonstrations the cost function is $J = J_{\text{target}} + J_{\text{effort}}$, while the LIPM demonstrations include the stepping cost, $J = J_{\text{target}} + J_{\text{effort}} + J_{\text{step}}$.

4.2 Numerical Results

4.2.1 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 prior to the control loop using the `MatlabFunction` code generator and symbolic toolbox. After each elapsed sample period, the states are sampled, the planning optimization 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.

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 ~ 7ms of computation time. Recent testing has further decreased this time to ~ 2ms (500Hz) using the OSQP solver [69].

4.2.2 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. 4.2A). 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. 4.2D. We further explore similar single-obstacle scenarios on drone hardware in Sec. 4.3.

4.2.3 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. 4.8), 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. 4.3). In some situations it would appear the agent (in blue) was trapped, but this simple heuristic method would find routes which encroached on the adversary's (in red) soft constraints (black circle). Sec. 4.2.5 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. 4.4). Using constraint planes to navigate around obstacles we demonstrate successful scaling to 3D in both performance and tractable computation.

4.2.4 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. 4.5) to the obstacle avoidance, where this heuristic setup is robust to a dynamic environments, and the agents do not collide with each other. The setup presented here uses individual QP's solved for each agent's trajectory.

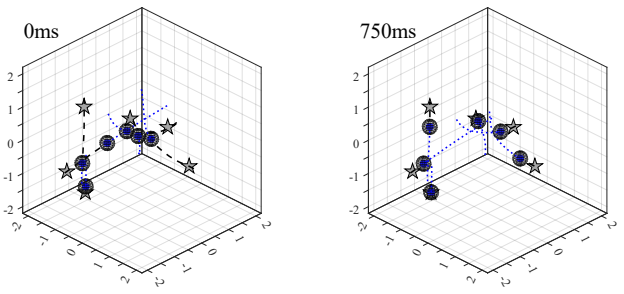

Figure 4.6: Simulated results of 3D uncoordinated multi-agent avoidance and waypoint tracking. Left image shows the initial plan to waypoints with black dashed lines, and the right shows 0.75 second into the simulation.

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, 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. 4.6B), again demonstrating the proposed method scales effectively to 3D for multiple non-cooperative agents.

4.2.5 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

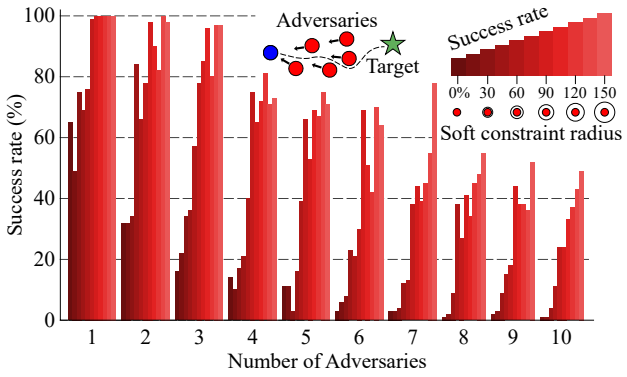

Figure 4.7: Reliability testing of multiagent avoidance in a tight corridor as a function of soft constraint radius.

10 obstacles. Starting at a position of $[4, 0]$, the agent is tasked to travel to $[0, 0]$, 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 velocity near zero, or collides with an obstacle. The corresponding success rates out of 100 trials are presented in (Fig. 4.7).

4.2.6 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. 4.8A). The robot stepping time T_{step} is 0.5s with a sample time ΔT 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. 4.8B) and with obstacle avoidance (Fig. 4.8C) 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

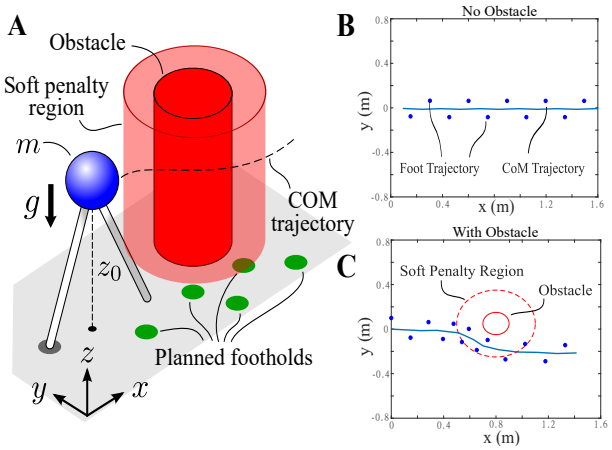

Figure 4.8: Bipedal motion planning in real-time using a Linear Inverted Pendulum Model (LIPM). **A.** Diagram for the numerical experimental setup. **B.** An MPC-generated motion without an obstacle. **C.** Real-time bipedal obstacle avoidance using the presented motion planner.

LIPM-MPC capabilities for more-complex geometries and dynamic environments.

4.3 Hardware Experiments

4.3.1 Experimental Setup

Hardware Description

Hardware experiments were carried out using a BitCraze quadrotor. The BitCraze quadrotor CrazyFlie uses brushed DC motors and are 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 infared 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 a number of other universities. The University of Southern California developed a software package called "Crazyswarm" [74] 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 [75]. It allows for steep roll/pitch angles, and develops minimum snap trajectories in real time using a sequence of 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.

4.3.2 Systematic Experiment: Pendulum Avoidance

To test the effectiveness of the "at risk" soft distance constraint, we incrementally changed the $\mathbf{p}_{risk}$ value for the quadrotor, and proceeded to swing a pendulum at it

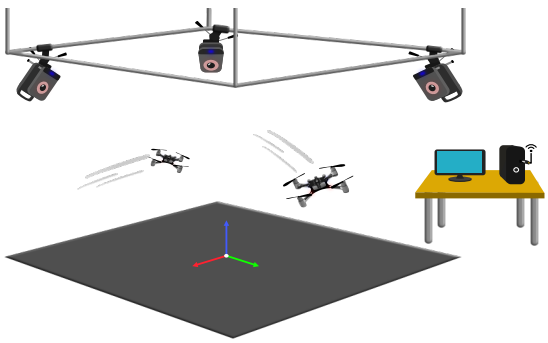

Figure 4.9: Lab experimental setup for quadrotors. Vicon cameras track the location of all agents and adversaries to provide sensory feedback.

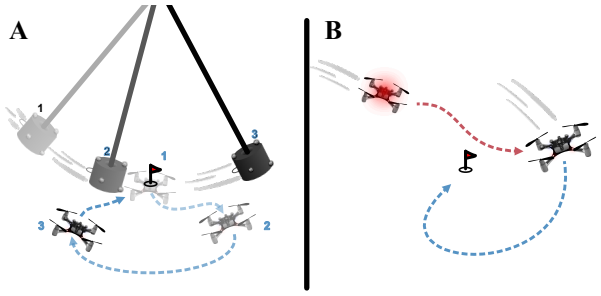

Figure 4.10: In our experiments, the quadrotor must minimize its distance to a waypoint while **A.** avoid a swinging pendulum and **B.** avoiding a chasing adversary quadrotor controlled by a PD chase law.

(3 times for each of the 7 radii), as seen in (Fig. 4.10A). 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. 4.11), 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**.

4.3.3 Demonstration: Adversarial Pursuit

We validate the numerical results from Section 4.2 by mimicking the adversary setup. As seen in Fig. 4.9B, 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.

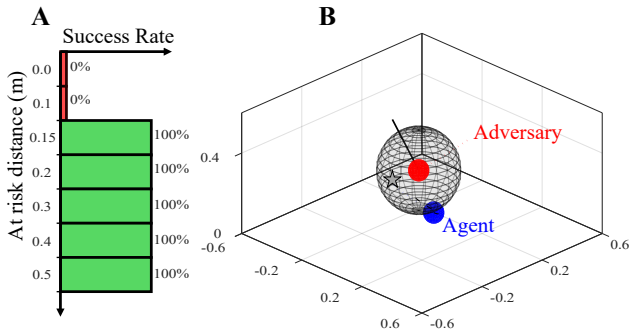

Figure 4.11: Pendulum experiment results. **A.** Soft penalty distance, d_{risk} and the avoidance success rate for 3 pendulum swing attempts. **B.** Example trajectory data from pendulum test.

2D Adversary Avoidance

Our initial adversarial experiments were constrained the avoidance to two dimensions. Representative experimental data can be seen in (Fig. 4.12B) 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. 4.12C). 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.

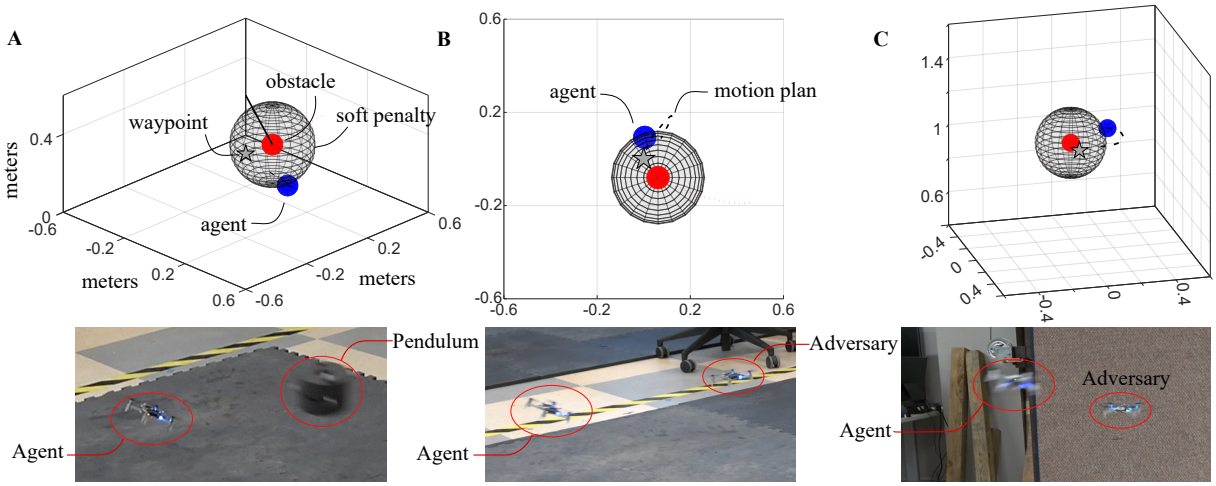

Figure 4.12: Experimental results trajectory data and example images. **A.** Quadrotor avoids a pendulum swinging at speeds up to 10m/s. **B.** The quadrotor is chased by an adversary drone and must avoid while staying within a 2D horizontal plane. **C.** Quadrotor adversary chase in 3D. See supplementary video for real-time footage.

4.4 Discussion

Avoidance Behavior

There was a clear correlation between the slack variable weight ($\mathbf{W}_{\text{avoid}}$), the soft constraint distance ($d_{soft} = \|\mathbf{p}_{risk} - \mathbf{p}_{col}\|_2$), 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 (d_{soft}) with a very large weight ($\mathbf{W}_{\text{avoid}}$) 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 is 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, are able to successfully avoid dynamic obstacles, which we believe speaks to the efficacy of the proposed half space method with soft constraints.

The avoidance capability of the motion planner on hardware is promising for future implementations and extensions. The half-space approach should extend to non-spherical obstacles by iteratively identifying the closest collision point. 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.

4.5 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.

References

[1] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, 2003, pp. 1620–1626 vol.2.

[2] R. B. McGhee and A. A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331–351, 1968.

[3] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[4] B. Katz, J. D. Carlo, and S. 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.

[5] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "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.

[6] M. Lecointe, C. Ponzoni Carvalho Chanel, and F. Defay, "Backstepping control law application to path tracking with an indoor quadrotor," 2015.

[7] P. Kokotovic, "The joy of feedback: nonlinear and adaptive," *IEEE Control Systems Magazine*, vol. 12, no. 3, pp. 7–17, 1992.

[8] A. Zulu and S. John, "A review of control algorithms for autonomous quadrotors," *arXiv preprint arXiv:1602.02622*, 2016.

[9] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[10] S. Sawyer, "Gain-scheduled control of a quadcopter uav," Master's thesis, University of Waterloo, 2015.

[11] R. I. Pérez, G. Galvan, A. Vázquez, S. Melo, and D. Alabazares, "Attitude control of a quadcopter using adaptive control technique," *Adaptive Robust Control Systems*, 2017.

[12] H. Razmi and S. Afshinfar, "Neural network-based adaptive sliding mode control design for position and attitude control of a quadrotor uav," *Aerospace Science and technology*, vol. 91, pp. 12–27, 2019.

[13] L. Doitsidis, K. P. Valavanis, N. C. Tsourveloudis, and M. Kontitsis, "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.

[14] M. Vukobratovic and D. Juricic, "Contribution to the synthesis of biped gait," *IEEE Transactions on Biomedical Engineering*, no. 1, pp. 1–6, 1969.

[15] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1620–1626.

[16] G. Wiedebach, S. Bertrand, T. Wu, L. Fiorio, S. McCrory, R. Griffin, F. Nori, and J. Pratt, "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.

[17] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the ATLAS humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[18] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim, "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.

[19] M. Shrivastava, A. Dutta, and A. Saxena, "Trajectory generation using GA for an 8 dof biped robot with deformation at the sole of the foot," *Journal of Intelligent and Robotic Systems*, vol. 49, no. 1, pp. 67–84, 2007.

[20] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: A scalable formulation for HZD gait optimization," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 370–387, 2018.

[21] X. Xiong, A. D. Ames, and D. 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.

[22] W. Bosworth, J. Whitney, S. Kim, and N. Hogan, "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.

[23] N. Rotella, S. Schaal, and L. Righetti, "Unsupervised contact learning for humanoid estimation and control," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 411–417.

[24] A. H. Chang, C. Hubicki, A. Ames, and P. A. Vela, "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.

[25] H.-j. Kang, K. Hashimoto, K. Nishikawa, E. Falotico, H.-o. Lim, A. Takanishi, C. Laschi, P. Dario, and A. Berthoz, "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.

[26] N. Paine, J. S. Mehling, J. Holley, N. A. Radford, G. Johnson, C.-L. Fok, and L. Sentis, "Actuator control for the NASA-JSC valkyrie humanoid robot: A decoupled dynamics approach for torque control of series elastic robots," *Journal of Field Robotics*, vol. 32, no. 3, pp. 378–396, 2015.

[27] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid *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] B. Henze, M. A. Roa, and C. Ott, "Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1522–1543, 2016.

[29] D. Kim, Y. Zhao, G. Thomas, B. R. Fernandez, and L. Sentis, "Stabilizing series-elastic point-foot bipeds using whole-body operational space control," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1362–1379, 2016.

[30] D. Kim, S. J. Jorgensen, H. Hwang, and L. Sentis, "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.

[31] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[32] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[33] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4170–4176, 2019.

[34] C. Hubicki, A. Abate, P. Clary, S. Rezazadeh, M. Jones, A. Peekema, J. Van Why, R. Domres, A. Wu, W. Martin *et al.*, "Walking and running with passive compliance: Lessons from engineering: A live demonstration of the ATRIAS biped," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 23–39, 2018.

[35] X. Da and J. Grizzle, "Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1063–1097, 2019.

[36] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," *arXiv preprint arXiv:1903.09537*, 2019.

[37] X. Xiong and A. 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.

[38] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *Proceedings of the American Control Conference*, 2019.

[39] K. Sreenath, H.-W. Park, I. Poulakakis, and J. W. Grizzle, "Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on mabel," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 324–345, 2013.

[40] A. Hereid, S. Kolathaya, and A. 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.

[41] C. Hubicki, A. Abate, P. Clary, S. Rezazadeh, M. Jones, A. Peekema, J. Van Why, R. Domres, A. Wu, W. Martin, H. Geyer, and J. Hurst, "Walking and running with passive compliance: Lessons from engineering: A live demonstration of the atrias biped," *IEEE Robotics Automation Magazine*, vol. 25, no. 3, pp. 23–39, 2018.

[42] W. J. Schwind, *Spring loaded inverted pendulum running: A plant model*. University of Michigan, 1998.

[43] A. Seyfarth, H. Geyer, M. Günther, and R. Blickhan, "A movement criterion for running," *Journal of biomechanics*, vol. 35, no. 5, pp. 649–655, 2002.

[44] M. Ernst, H. Geyer, and R. Blickhan, "Extension and customization of self-stability control in compliant legged systems," *Bioinspiration & biomimetics*, vol. 7, no. 4, p. 046002, 2012.

[45] B. Andrews, B. Miller, J. Schmitt, and J. E. Clark, "Running over unknown rough terrain with a one-legged planar robot," *Bioinspiration & biomimetics*, vol. 6, no. 2, p. 026009, 2011.

[46] G. Piovan and K. Byl, "Reachability-based control for the active slip model," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 270–287, 2015.

[47] M. Calisti and C. Laschi, "Underwater running on uneven terrain," in *OCEANS 2015-Genova*. IEEE, 2015, pp. 1–5.

[48] G. Picardi, C. Laschi, and M. Calisti, "Model-based open loop control of a multigait legged underwater robot," *Mechatronics*, vol. 55, pp. 162–170, 2018.

[49] R. Alicea, K. Ladyko, and J. Clark, "Lift your leg: Mechanics of running through fluids," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7455–7461.

[50] M. P. Austin and J. 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.

[51] W. Xi, Y. Yesilevskiy, and C. D. Remy, "Selecting gaits for economical locomotion of legged robots," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1140–1154, 2016.

[52] O. V. Stryk, "Numerical solution of optimal control problems by direct collocation," in *Optimal control*. Springer, 1993, pp. 129–143.

[53] M. S. Jones, "Optimal control of an underactuated bipedal robot," 2014.

[54] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

[55] C. Li, T. Zhang, and D. I. Goldman, "A terradynamics of legged locomotion on granular media," *Science*, vol. 339, no. 6126, pp. 1408–1412, 2013.

[56] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[57] J. V. D. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[58] D. Bareiss and J. Van Den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[59] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "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.

[60] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[61] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 773–11 780, 2014.

[62] T. J. Stastny, A. Dash, and R. Siegwart, "Nonlinear mpc for fixed-wing uav trajectory tracking: Implementation and flight experiments," in *AIAA guidance, navigation, and control conference*, 2017, p. 1512.

[63] M. J. Powell, E. A. Cousineau, and A. 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.

[64] H. Cheng, Q. Zhu, Z. Liu, T. Xu, and L. Lin, "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.

[65] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[66] M. Szmuk, C. A. Pascucci, D. Dueri, and B. Açikmeşe, "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4862–4868.

[67] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Nonlinear model predictive control for multi-micro aerial vehicle robust collision avoidance," *arXiv preprint arXiv:1703.01164*, 2017.

[68] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using panoc," in *2018 European control conference (ECC)*. IEEE, 2018, pp. 1523–1528.

[69] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[70] Y. Chen, S. Huang, and R. Fitch, "Active slam for mobile robots with area coverage and obstacle avoidance," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 3, pp. 1182–1192, 2020.

[71] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2001, pp. 239–246.

[72] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[73] K. Wang, H. Fei, and P. Kormushev, "Fast online optimization for terrain-blind bipedal robot walking with a decoupled actuated slip model," 2021.

[74] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304.

[75] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.