Towards Safe Autonomy in Assistive Robots

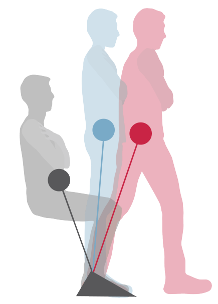
by

Patrick Daniel Holmes

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2021

Doctoral Committee:

Associate Professor Ramanarayan Vasudevan, Chair
Associate Professor Robert D. Gregg
Assistant Professor Talia Y. Moore
Assistant Professor Elliott J. Rouse



Patrick Daniel Holmes
pdholmes@umich.edu
ORCID iD: 0000-0001-5252-4664

© Patrick Daniel Holmes 2021

DEDICATION

Violet and green,
a hummingbird's wings beat on
the same air I breathe.

In memory of Alexandra Joyce Tang.

ACKNOWLEDGMENTS

I would not be able to earn a PhD without a community of strong supporters and a wealth of amazing opportunities for which I must give thanks. First, I would like to thank my advisor Ram Vasudevan and committee members Talia Moore, Bobby Gregg, and Elliott Rouse.

Ram, you have been an incredible advisor. You are gracious, patient, accessible, and have always tried to put me on the right path towards my goals. You bring a warmth and curiosity that makes it a joy to work with you, and you care deeply about each of your students. You have been relentlessly supportive of me throughout my time here, and have believed in me more than I believed in myself. As a researcher and mentor, you are a daunting role model to aspire to, but it is your strength of character and humility that I find most inspiring. I am so grateful that you accepted me as your student, and thankful for all the time and effort you have put into teaching and guiding me throughout my PhD.

Talia, thank you for taking me under your wing and being such an important mentor to me. Your wonder and appreciation for the natural world is inspiring, and you encourage me to think more broadly about the role robots can play in our future. I admire how you consistently approach people with unmatched kindness and care. You taught me how to think like a scientist and how to effectively communicate my research through your meticulous eye for detail. Throughout my PhD, you have provided a unique perspective on my work and led me to craft stronger and more consistent arguments. I love listening to you tell stories and share interesting facts, and I am continually impressed by your deep and broad knowledge across a wide variety of topics. Thank you so much for coaching me and always offering helpful suggestions for how to improve my work.

Bobby and Elliott, thank you for your helpful feedback, mentorship, and time. You both do work that I find fascinating, and I am excited to hopefully work more closely with you in the new Robotics building. I strive to emulate your passion for building and controlling robots that make a direct and positive impact on a person's life, and for more fundamentally understanding how human motion is constructed. A special thanks to Elliott, who was incredibly accommodating to my dissertation schedule despite a medical leave and provided important suggestions that strengthened this work.

Next, I would like to thank my labmates in ROAHM Lab. Roughly in the order I met you, thanks to Shreyas, Shankar, Pengcheng, Henry, Dan, Josh, Nils, Shannon, Sean, Matt, Jinsun, Ming-Yuan, Mani, Fan, Owen, Basel, Hyongju, Utkarsh, Ali, Frank, Tapan, Daphna, Aohan, Arnav, Zhenyu, Sid, Hannah, Jon, Zehui, Xun, Bohao, Challen, Zac, Miracle, Simon, Martin, Parker, Lucas, Yiting, Daniel, Callie, and Steven. I have had so much fun working with you and alongside you, and am inspired by the talent and diligence that you all bring to your work. They say it takes a village, and you were the village! Somehow you made a windowless basement in G.G. Brown feel like home, and now we have so many windows.

I especially want to thank Shannon and Shreyas, two of my labmates that made the work in this dissertation possible. Shannon, I am excited to go to lab every day to work on human motion with you, even days where we just grimace emoji at each other. Your optimism and perseverance are exceptional, and you are unafraid to tackle extremely hard problems. Thank you for always being there to help me think through things, or just to show you a figure or animation I am overly proud of. Many of my favorite memories during my PhD are from trips to conferences with you; thanks for being such a great travel buddy! Turns out you are also a genius at crafting educational music videos, and I can not wait for our next hit.

Shreyas, I did not want to touch robots until you convinced me that it would be fun to work together on them. I learned so much while playing with quadrotors and developing ARMTD with you, and it broadened my perspective on what I wanted my work to do. You bring so much joy to your work and revel in the math, writing, and organizational details I abhor. I loved drawing blobs on the whiteboard that probably only made sense to us. I am so glad that you were the person I shared every part of grad school with from the very start.

I want to give a special thanks to Xiao-Yu Fu. I could not have completed the Stability Basin work without your help with the experiment, and you have been an important source of knowledge and experience with human biomechanics. I would also like to thank Camila Shirota and Levi Hargrove, who generously made the data used for testing TRIP-RTD accessible to us.

Next, I would like to thank my family. Pa, I have wanted to get a PhD since I was a little kid so that someday I could be as smart as you. I still want to be just like you when I grow up. Mom, thank you for always being proud of me and taking care of me (and always being right). You never pushed me to succeed, but you always *expected* me to because you believed in me. Michael, you work hard at everything that you do and that inspires me every day. Thank you for always taking your little brother with you wherever you went. Camtu, you have been like a big sister to me. I can not wait to meet the newest member of our family!

Thanks to my extended family for all the love and encouragement over the years. My grandparents, aunts, uncles, and cousins have always been a huge part of my life, and I am continually grateful for their support.

I want to give a big thank you to my friends, especially my Munger friend group and those I have met at Michigan; you all became my family here.

Finally, I would like to thank my partner, Victoria. You are the first person I turn to about anything, and your love and support has been critical to me finishing this work. I am so thankful to be loved by someone who knows me inside and out. You approach life with joy and a passion for what is right, and you inspire me to be the best person that I can be. I love you, and am so excited to start the next chapter of our lives together.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TABLE

ABSTRACT

Robots have the potential to support older adults and persons with disabilities on a direct and personal level. For example, a wearable robot may help a person stand up from a chair, or a robotic manipulator may aid a person with meal preparation and housework. Assistive robots can autonomously make decisions about how best to support a person. However, this autonomy is potentially dangerous; robots can cause collisions or falls which may lead to serious injury. Therefore, guaranteeing that assistive robots operate safely is imperative.

This dissertation advances safe autonomy in assistive robots by developing a suite of tools for the tasks of *perception, monitoring, manipulation* and *fall prevention*. Each tool provides a theoretical guarantee of its correct performance, adding a necessary layer of trust and protection when deploying assistive robots. The topic of interaction, or how a human responds to the decisions made by assistive robots, is left for future work.

Perception: Assistive robots must accurately perceive the 3D position of a person's body to avoid collisions and build predictive models of how a person moves. This dissertation formulates the problem of 3D pose estimation from multi-view 2D pose estimates as a sum-of-squares optimization problem. Sparsity is leveraged to efficiently solve the problem, which includes explicit constraints on the link lengths connecting any two joints. The method certifies the global optimality of its solutions over 99% of the time, and matches or exceeds state-of-the-art accuracy while requiring less computation time and no 3D training data.

Monitoring: Assistive robots may mitigate fall risk by monitoring changes to a person's stability over time and predicting instabilities in real time. This dissertation presents Stability Basins which characterize stability during human motion, with a focus on sit-to-stand. An 11-person experiment was conducted in which subjects were pulled by motor-driven cables as they stood from a chair. Stability Basins correctly predicted instability (stepping or sitting) versus task success with over 90% accuracy across three distinct sit-to-stand strategies.

Manipulation: Robotic manipulators can support many common activities like feeding, dressing, and cleaning. This dissertation details ARMTD (Autonomous Reachability-based Manipulator Trajectory Design) for receding-horizon planning of collision-free manipulator trajectories. ARMTD composes reachable sets of the manipulator through workspace from low dimensional trajectories of each joint. ARMTD creates strict collision-avoidance constraints from these sets, which are enforced within an online trajectory optimization. The method is demonstrated for real-time planning in simulation and on hardware on a Fetch Mobile Manipulator robot, where it never causes a collision.

Fall Prevention: Wearable robots may prevent falls by quickly reacting when a user trips or slips. This dissertation presents TRIP-RTD (Trip Recovery in Prostheses via Reachability-based Trajectory Design), which extends the ARMTD framework to robotic prosthetic legs. TRIP-RTD uses predictions of a person's response to a trip to plan recovery trajectories of a prosthetic leg. TRIP-RTD creates constraints for an online trajectory optimization which ensure the prosthetic foot is placed correctly across a range of plausible human responses. The approach is demonstrated in simulation using data of non-amputee subjects being tripped.

CHAPTER 1

Introduction

The share of older persons as a percentage of the global population is rapidly increasing. In 2015, older persons – those aged 60 years or over – comprised $1/8^{\text{th}}$ of the global population according to the United Nations [6]. By 2030, this share is expected to increase to $1/6^{\text{th}}$ of the population. These demographic shifts are the result of declining birth rates and improved longevity.

An aging population presents the socio-demographic challenge of ensuring "successful" aging. Though successful aging is broadly defined, it depends on a person's physical and cognitive performance and disability, as well as their independence, social relationships, and resilience to adversity [7, 8]. These interdependent factors are strongly influenced by a person's **mobility**. Broadly, mobility is defined as

> "the ability to move oneself (e.g., by walking, by using assistive devices, or by using transportation) within community environments that expand from one's home, to the neighborhood, and to regions beyond." [9].

Age-related changes to mobility may occur as one facet of the heterogeneous and complex process of aging [10], for example due to the impairment of vision or the loss of muscle mass. Though the impact of changes to mobility depends heavily on the individual and their living situation, in general impaired mobility is correlated with morbidity, loss of independence, and reduced quality of life [11, 12].

Personal mobility is required to perform many activities of daily living (ADL), such as eating, walking, dressing, or using the toilet [13]. Additionally, many instrumental activities of daily living (IADL), such as cooking, shopping, doing laundry and using transportation, are more complex and may require a higher level of personal mobility [14]. Devices such as walkers and powered wheelchairs can support a person's mobility needs [15], and a person's environment may be altered to support ADL, for example by adding handrails in the bathroom. However, for some older persons, caregivers provide essential support in carrying out ADL and IADL. These may either be formal caregivers, as in a retirement community or hospital, or informal caregivers, such as spouses, children, or other friends and family members. Informal caregivers play an important role in enabling “aging-in-place”, with a majority of older persons preferring to be cared for at home rather than institutional settings [16]. Caregivers can provide physical assistance for ADL by helping a person get dressed or offering support going down the stairs [17]. They may also provide assistance with IADL by shopping, cooking, or doing household tasks. While many informal caregivers report mainly positive attitudes towards caring for loved ones, providing care can also be physically and mentally taxing [18]. This may lead to fatigue and stress depending on the level of dependency of the older person and on the presence of perceived social support [19].

Assistive robots have been suggested as a means to support older persons and persons with disability during daily tasks [20, 21, 22, 23]. Broadly, an assistive robot can be defined as

> “an actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform tasks and services which are used by disabled and/or elderly people to overcome social, infrastructural and other barriers to independence, full participation in society and carrying out activities safely and easily.” [24]

The roles that these assistive robots could play are expansive and varied. Assistive robots could promote personal mobility by offering physical support during walking, balancing, and dynamic tasks like sit-to-stand [25]. Assistive robots could also help with cooking and cleaning, fetching items within the home, and act as social companions. By offering additional support to older persons when necessary, assistive robots may allow for greater independence and increased access to personalized care [10]. Consequently, this may reduce the amount of support and effort required of formal and informal caregivers.

1.1 The Role of Assistive Robots

1.1.1 Personal Mobility Needs

This dissertation focuses on “physically assistive robots” which augment personal mobility. The dissertation does not consider “socially assistive robots” which may support a person’s cognitive and social needs, although significant overlap between these types of robots is possible [24]. The role that physically assistive robots may play in an older person’s life depends on their personal mobility needs. Fiorini et al. identified four categories of mobility needs that could be addressed by assistive robots by interviewing older persons as well as formal and informal caregivers [25]. We recontextualize these needs and explore the functions that assistive robots may perform to meet these needs.

1. The first are *physical needs* which correspond to ADL, such as getting up from a chair or walking inside the house. Assistive robots can physically interact with a person as they perform these tasks. This support could be passive (such as acting as a portable handhold [26]) or active by aiding with balance during walking [27] or providing lifting force when getting up from a chair or bed [28]. Additionally, caregivers suggested that such robots could help them lift or carry older persons, or help turn them over in their bed, which could ease the physical strain of their work.

2. The second are *instrumental needs*, which correspond to IADL such as "being able to fetch and carry items" and "doing housework". Assistive robots equipped with manipulator arms can autonomously complete tasks within this domain, such as fetching a cup of coffee, changing cat litter [10], or putting away dishes in the kitchen [29].

3. The third are *rehabilitation needs*, which includes "doing physical exercise aimed at maintaining or improving mobility." Older adults suggest assistive robots may be useful for motivating them to perform these exercises and for assessing their performance. Caregivers also believe robots could automatically assess gait and other measures to determine how to minimize fall risk. In this regard, special care should be taken to ensure the safety of personal data; robots should avoid the negative perception of "always watching" or surveilling a person by allowing a person to choose how their data is collected and managed [24].

4. The fourth are *personal safety needs*, which means minimizing fall risk and being able to alert a caregiver in case of a fall. If an older person is living alone, assistive robots may automatically detect if a fall has occurred and alert emergency services if necessary [26]. Assistive robots may actively seek to prevent falls, for example by detecting and removing obstacles from the floor. Wearable assistive robots may augment a user's reaction to a trip or slip to prevent falls by automatically moving a person's limbs in response to instability [30].

1.1.2 Technical and Functional Requirements

These needs are complex and motivate the design of assistive robotic systems which can meet one or more of these needs simultaneously. To achieve the types of support suggested above, assistive robots must meet a plethora of technical and functional requirements:

- *Perception.* First, assistive robots must use sensors to understand their environment and the people within it. This includes building maps of the environment for autonomous navigation and understanding the location of important objects in the environment. Furthermore, assistive robots should be able to localize people within this environment and to predict their future motions to avoid potential collisions.

- *Motion and Navigation.* Assistive robots need to move within the long-term care environment to perform useful tasks. This may involve autonomously navigating between rooms and moving in tight spaces. Wearable assistive robots may physically support a person by moving their limbs or maintaining balance. These robots may be personalized to assist only when necessary based on a user's strength [31].

- *Monitoring.* Assistive robots may collect and analyze large amounts of user data because they are equipped with a suite of mobile sensors [32]. This could prove vital for detecting anomalous events such as falls, or for predicting when a person would like support for moving from one place to another. Furthermore, assistive robots could monitor a person's risk of falling by assessing changes to a person's performance of a task over time, such as the onset of gait impairment.

- *Manipulation.* Many daily tasks around the home require the grasping and transportation of objects. These include cooking, feeding, cleaning, dressing, and doing the laundry [29, 33]. Assistive robots must be equipped with manipulation capabilities to perform these tasks. Manipulation for these tasks is significantly more challenging than traditional applications of robotic manipulation, such as factory assembly lines, because the home environment is much less structured and robots may need to manipulate objects in close proximity to humans.

- *Interaction.* Understanding a person's needs and intent from sensor data is a difficult but essential function of assistive robots [34]. Interfacing with these robots should be simple and intuitive to understand, such as through voice and gesture recognition [35]. Ideally, robots should be able to recognize a user's emotions or mood when deciding how best to support them.

- *Fall Prevention.* Because falls are a grave concern for older persons, assistive robots should move in ways which actively seek to minimize fall risk [22]. For example, this may mean positioning themselves as a mobile handhold to support gait or sit-to-stand transitions. Wearable robots may prove especially useful by responding to external perturbations such as trips or slips and compensating for slow reaction times or muscle weakness.

1.1.3 Types of Assistive Robots

Researchers have proposed and developed numerous assistive robots which incorporate the functionality described in the previous section. We provide a brief overview of existing research platforms and commercial products and discuss their capabilities.

- *Service robots* that have been proposed are generally comprised of a wheeled mobile base, sensors (like a Microsoft Kinect) for perception, and may take forms suggestive of humans. These include the MARIO/Kompai robot [36], Care-o-Bot 4 [37], DoRo [38], RAMCIP robot [39], and ASTRO robot [26]. Some robots, such as the Care-o-Bot 4 [37], are capable of expressing "emotions" through the inclusion of expressive eyes. A tablet computer attached to the robot often serves as the primary means of interfacing with the robot. These robots may foster social inclusion, provide a means for telepresence, remind persons to take their medication, or alert caregivers if a fall is detected. Generally, the impact of these robots on supporting personal mobility is limited. However, they may be capable of manipulation tasks, such as the RAMCIP robot [39] which can manipulate light switches and pick and place objects. Furthermore, the ASTRO robot [26] may assist with sit-to-stand by providing a mobile handhold which a person may grasp to pull themselves up a seated position.

- *Manipulators* are robotic arms that may be attached to a mobile base and assist with common IADL, like fetching and carrying items and doing housework. Service robots, such as the RAMCIP robot [39] and Care-o-Bot 4 [37], may use attached manipulators while assisting users. Mobile manipulator platforms such as Willow Garage's PR2 [40] and the Fetch Mobile Manipulator robot [41] may serve as assistive robots in addition to other roles for which they have been imagined. The Toyota Research Institute is developing a ceiling-mounted manipulator for use in a home's kitchen. This manipulator may slide on a track located on the ceiling, and complete common tasks such as cleaning the counters, opening the refrigerator, and grasping cups and bottles [29]. A series of robotic manipulators were developed to assist with vocational activities at a user's office workstation (the MoVAR, DeVAR [42], and ProVAR [43]), which included a GUI and virtual representation of the workspace and achieved common tasks like handling computer disks.

  Many wheelchair-mounted manipulators have been developed to support users with limited upper-body mobility. These include the KARES I [44] and KARES II [45] robots and the MANUS arm [46]. More recently, the commercially available Kinova Jaco arm has shown impressive manipulation capabilities [47]. A user reported being able to open their residence's mailbox with a key and retrieve the mail using this arm [48]. The arm is primarily controlled by a joystick or other user interface; increasing the autonomy of such arms may improve the speed and accuracy of assistance.

- *Wearable robots* provide physical support to a user through direct interaction with a person's body. These include powered exoskeletons and powered prosthetic limbs.

  In recent years, numerous companies have been created which develop powered exoskeletons [49, 50]. These exoskeletons have accomplished the amazing feat of robot-assisted walking for patients with complete spinal cord injuries [51]. Research is also being conducted for exoskeletons to assist caregivers in moving patients [28]. Studies have shown that exoskeletons may be used to prevent falls by detecting and responding to instabilities [30]. The current development of lightweight soft exosuits may lead to broader adoption of these technologies in older persons [52], and could provide personalized assistance with tasks like stair climbing [53].

Powered prosthetic legs, in contrast to conventional prostheses, can supply energy during locomotion via knee and ankle motors [54]. This can reduce the metabolic cost of walking [55] and promote gait symmetry [56]. Recent advances in the design of these robotic legs have led to powerful, lightweight and low cost systems [57]. In addition to level-ground walking, these devices may assist users with other tasks like walking up slopes, ascending/descending stairs, and walking backwards [58].

- *Powerchairs & walkers* may extend the range a person may travel independently while decreasing risk of falls. Though typically controlled by a joystick (powerchairs) or manually transported (walkers), researchers have studied increasing the autonomy of these systems. For example, robotic powerchairs may share control with users to reduce the amount of user input needed for operation, or robotic walkers may move themselves while staying in front of a person to reduce the work done by a user [59, 15]. Researchers have studied the use of robotic walkers for monitoring and analyzing a person's gait via laser range finders [27].

- *Socially assistive robots* have been suggested to support older persons' social and cognitive needs [60, 61, 62, 63]. Though this dissertation focuses on personal mobility needs, we include a brief discussion of such robots here.

  A popular example is the PARO robot. PARO is an animatronic seal used to support patients with dementia [64]. The robot was found to reduce negative emotion and improve positive mood and quality of care [65]. Socially assistive robots are also being extensively researched for treating autism in children, where they have been found to increase user therapy acceptance and improve social skills [66]. One theory for the positive reception these robots often receive among these children is that they offer predictable responses to the child's actions [67]. These responses may be designed to become more complex over time to bolster the child's social skills.

1.2 Safe Autonomy

Underpinning the functional requirements presented in Sec. 1.1.2 are two central concepts of this dissertation: safety and autonomy. **Autonomy** refers to a robot's ability to sense and interpret people and the environment, and to independently make decisions and take actions to assist users. Increasing autonomy can relieve users from having to constantly specify the desired operation of these robots. For example, a robotic walker may choose its own path through a crowded environment [68]. As another example, if a person asked a robot to fetch a cup of water, it may be unreasonable to expect a person to control each of the robot's joints as it maneuvered to and grasped the cup [69]. Instead the robot should be able to interpret the person's high level intentions and translate these into actions that complete a given task. Allowing assistive robots to choose their own actions may make them more useful; however, it opens the door for robots to choose actions which are not **safe**.

What it means for a robot to "move safely" depends on context and the nature of their interactions with humans [70]. In this dissertation, we focus on the physical safety of the person and consider a robot's motion unsafe if it could cause a person to fall (as in the case of wearable robots) or cause a collision with the person (as in the case of manipulators). Safety is doubly important when considering that older adults and persons with disabilities may be more at risk of severe injury from a fall or unexpected collision [71, 72]. We make the argument for **safe autonomy** through the following line of reasoning:

1. **Assistive robots can be dangerous** and may put humans at serious risk of injury. Some of the service robots described in Sec. 1.1.3 such as the ASTRO robot [26] can be as tall as a person and weigh over a hundred pounds. Collisions between these robots and an older person could knock them off balance and cause a fall which could lead to severe injury or death. This danger is more pronounced for wearable robots, which are constantly physically interacting with a person's body. An unexpected motion autonomously planned by such robots could similarly cause a devastating fall (e.g. by causing the user to trip on an obstacle).

   Manipulators present another source of danger; in some tasks such as feeding, robot arms must come into extremely close contact with a person [73]. These stiff, heavy arms may cause injury through collision with a person, especially if moving at a higher speed. These dangers are especially pronounced when manipulating sharp objects like forks or knives. Manipulators operating away from a person may still cause hazards through unwanted collisions with the environment (e.g. by knocking a glass of water onto the floor and creating a slip hazard).

2. **Guarantees of safety are necessary.** Some level of risk will always be present when robots

operate autonomously near humans. However, roboticists can safeguard against potential hazards through careful design.

The first solution is to ensure that robots and humans never occupy the same physical space. For example, large industrial robots on factory floors are caged off to prevent potential injury [74]. This is infeasible for assistive robots, which will operate in a person's home and may require physical contact with a person to perform certain functions.

Another possible solution is to design the physical morphology of the robot with safety in mind. The field of soft robotics has received much attention in recent years, with this property often cited as an advantage over traditional robots [75]. Soft robots are made from compliant, deformable materials, so collisions between these robots and humans are inherently much safer [76]. Unfortunately the control of soft robots is a formidable challenge. Soft robotic manipulators thus far can not match the functionality of traditional robotic manipulators composed of rigid links. However, researchers have recently demonstrated end-effector positioning [77, 78, 79] and pick and place functionality [80].

Padding the exterior of the manipulators and service robots described in Sec. 1.1.3 may make impacts with these robots somewhat less dangerous. However, they may still cause falls, knock over objects in the environment, or cause injury when manipulating sharp or heavy objects. Additionally, these arguments do not apply to wearable robots, where falls rather than collisions are the main source of danger. Therefore, it is insufficient to only consider the physical morphologies of these robots when attempting to guarantee robot safety.

A third solution is to control the robot in a way that guarantees unsafe actions can not occur, which we discuss subsequently.

3. **These guarantees often take the form of a mathematical proof.** Robot autonomy involves perceiving people and the environment, making predictions about agents' intentions, and deciding on and implementing a course of action. Algorithms at each step of this pipeline enable autonomous operation of the robot. *To make guarantees of safe autonomy, we must certify the correct performance of these underlying algorithms.* Additionally, these algorithms should be robust to uncertainty because a robot's models of itself and its environment will never be perfect.

   At their core, each of these algorithms is a function that transforms inputs (e.g. obstacle locations) into outputs (e.g. a collision-free motion plan). By developing mathematical proofs to certify that the outputs of these algorithms satisfy definitions of safety, we add a necessary layer of trust and protection when deploying assistive robots in a person's home.

1.3 Overview

1.3.1 Problem Statement

The goal of this dissertation is to enable safe autonomy in assistive robots by developing a suite of tools for **perception, monitoring, manipulation**, and **fall prevention**, which provide mathematical guarantees for their correct performance. These functions, introduced in Sec. 1.1.2, allow assistive robots to support the personal mobility needs discussed Sec. 1.1.1. This dissertation considers the following open problems for each of these functions:

1. **Perception:** Given 2D estimates of human pose from multiple RGB cameras, find the globally-optimal 3D reconstruction that minimizes measurement error while obeying realistic constraints on motion.

2. **Monitoring:** Given kinematic observations of human motion, build dynamic models which characterize a person's stability during motion and predict when a person will become unstable.

3. **Manipulation:** Given a set of obstacles and a robotic arm model, construct a real-time receding-horizon planner which finds the best robot action subject to strict collision-avoidance constraints.

4. **Fall prevention:** Given a set of predicted human responses to a stumble, plan trajectories of a wearable robot which guarantee that a fall will not occur.

The concept of *interaction*, or how a person responds to decisions made by assistive robots, will be critical to the successful deployment of these autonomous systems in a person's home. However, we do not explicitly examine the topic in this dissertation, instead leaving it for future work.

We use the following motivating example to introduce some of the challenges and motivate the open problems addressed by this dissertation.

1.3.2 Motivating Example

We consider the hypothetical situation of an older person at home in their kitchen assisted by several robots, displayed in Fig. 1.1. The first robot (Fig. 1.1(a)), a ceiling-mounted robotic arm [29], assists the person with moving plates from the dishwasher to the cupboard. The second robot, a wearable lower-limb exoskeleton [50] (Fig. 1.1(b)), assists the person with their balance and reduces the load on their muscles and joints during dynamic activities, like walking or standing up from a chair.

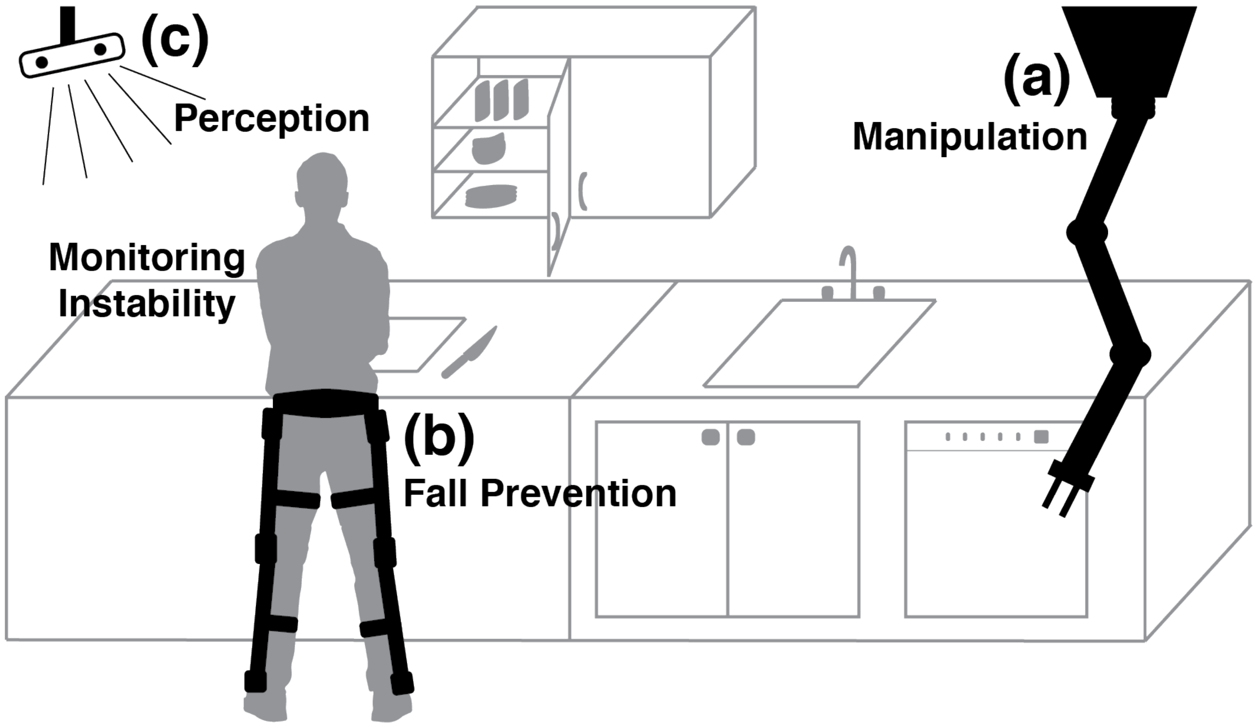


Figure 1.1: A depiction of our motivating example of a person in their kitchen assisted by several robots. (a) A ceiling mounted robotic manipulator helps with tasks around the kitchen such as moving dishes to and from the dishwasher. For this robot to be safe it must not cause a collision with the person (or unintentionally hit any other part of the kitchen). (b) A wearable robotic exoskeleton helps stabilize the person and assist with dynamic motions. For this robot to be safe it must not cause the person to fall and should actively intervene to keep the person from falling when appropriate. (c) External sensors such as ceiling mounted cameras may aid in localizing the person and estimating their movement. In this depiction a stereo camera perceives the person's 3D pose and communicates its estimates to both robots for use in planning. These estimates may be used to monitor the stability of the person over time and to identify potential instabilities in real time.

1. **Perception.** In this scenario, we consider the possibility of using several calibrated RGB cameras located either externally or on a robot platform for estimating the current 3D pose of the person as shown in Fig. 1.1(c). Correctly perceiving the position of the person's body within the kitchen is essential. For example, this is necessary for the arm to plan trajectories which avoid collision with the person. Estimates of the person's joint angles and center of mass state derived from vision-based perception can also be used for controlling the wearable robot.

2. **Monitoring.** Assistive robots can monitor how older persons complete dynamic tasks over time to identify changes in their stability [25]. In this scenario, perceptual estimates provided by the RGB cameras or encoders in the wearable robots joints' may provide valuable data for assessing the person's biomechanics over time. For example, data collected on a person's gait or common ADLs like sit-to-stand could be used to quantify their stability and predict their risk of falling. Such measures could also be used to detect instabilities in real-time, for example if the person tripped on a kitchen chair leg. In this case the wearable robot could take action to attempt to prevent a fall. Care must be taken to empower the person with control over how their data is used to allay fears of surveillance [24].

3. **Manipulation.** For the robotic arm in this scenario to move safely it must not cause a collision with any part of the person. Doing so could be catastrophic, for example if the person is chopping vegetables with a sharp knife. The robot arm should also not collide with any unintended part of the kitchen. The manipulator relies on estimates of the person's position when planning its path, and therefore it is critical that these estimates are accurate and reliable. Furthermore, the manipulator should be capable of replanning its motion several times per second to incorporate updated predictions of the person's movement.

4. **Fall Prevention.** Falls are the leading cause of injury in people over 75 years old, resulting in reduced quality of life, increased healthcare costs, and accident-related death [71, 81, 82]. Assistive robots may detect instabilities (caused by trips or slips) that could lead to a fall while *monitoring* the older person's motion. The exoskeleton worn by the person may intervene to prevent potential falls [30] by moving the person's joints. For example, the exoskeleton may assist in quickly placing the person's foot in a location where they can regain their balance.

   While it would be terrible for a wearable robot to fail to prevent a fall, it would be even worse for the robot to *cause* a fall by trying to avoid one. Therefore, the robot should be certain that intervention is necessary and that a planned recovery motion is safe.

1.3.3 Challenges

Algorithms for these functions of *perception, monitoring, manipulation* and *fall prevention* are united by several shared challenges:

1. **Difficult optimization problems:** Algorithms for these functions are often formulated as (or can be interpreted as) optimization problems. For example, finding the best trajectory for a robot arm while avoiding collisions [2], or finding the best estimate of a person's 3D pose given noisy sensor data [4]. Due to the high-dimensional nature of the robots in consideration, and the many degrees of freedom a person has, these optimization problems may have a large number of decision variables. The problems may also be nonlinear and non-convex, often requiring good initial guesses and lacking a guarantee that the solver will converge to a good solution.

2. **Theoretical guarantees:** Because the consequences of performing an unsafe action can be catastrophic it is important to certify that outputs of these algorithms satisfy certain desirable properties, as described in Sec. 1.2. In this dissertation we focus on the properties of safety (collision and fall avoidance) and optimality. For example, for *manipulation* we may require a theoretical proof that a planning algorithm is guaranteed not to cause a collision, and for *perception* we may enumerate conditions under which the output of an algorithm is guaranteed to be the best possible solution.

    a. Safety: It is important that robots certify the safety of their plans before executing them. This consists of choosing a plan that is guaranteed to be safe across a set of predicted human actions which, is assumed to contain the actual human action [83]. Ensuring safety is challenging and often comes at the cost of excessive conservatism or computational requirements. For example, if a robot arm checks for collision at a discrete number of configurations, it must either buffer its volume (adding conservatism) or discretize finer (adding computation time) to account for the times in between these discrete configurations [84]. One strategy for reducing computational requirements is to relax the safety constraint by placing a high (rather than infinite) penalty on violating the constraint [2, 85]. However, doing so requires post-processing of a solution to verify its safety.

    b. Optimality: Aside from safety it is also important that algorithms for these subproblems find the best possible solutions [86]. For example, when perceiving a person's position using noisy sensors, the true position of the person may be unknowable. However, producing the best possible estimate from noisy data gives subsequent prediction and planning algorithms the best chance at maintaining safety. Generally, it is difficult to know whether or not a better estimate exists. Certain relaxations of nonlinear, nonconvex optimization problems can provide guarantees of global optimality [87, 88], but in the worst case may require solving an infinite-dimensional program.

3. **Real-time computation constraints:** Because humans are constantly moving and a robot's environment is changing, it is vital that robots sense and plan their actions in real time [89]. Doing so places limits on the computation time a robot has to complete each of its functions. Obtaining solutions with guarantees of safety/optimality may already be computationally intractable, and the added challenges of limited time and computational resources only make these problems more difficult.

   One strategy to alleviate the burden of real-time computation is to "precompute" relevant objects ahead of time and to efficiently utilize them online. For example, when predicting instability, a robot could precompute a set of "stable" trajectories for a dynamic task ahead of time, then compare the person's motion against this precomputed set online. For planning, one can precompute sets of robot trajectories [90] or configurations [91] and attempt to quickly certify their safety online. However, it may be difficult to precompute objects which account for the full range of possible behaviors because a person or robot's next motion may depend heavily on an initial condition in some high-dimensional space.

1.4 Contributions

This dissertation develops tools for each function separately with the goal of building a unified framework for safe autonomy in assistive robots. Special attention is paid to the challenges cited in Sec. 1.3.2, with a focus on formulating tractable optimization problems, providing guarantees of safety and/or optimality, and accounting for real-time computation constraints. To address each of these problems, the dissertation is broken into four contributions. These contributions are contextualized within the schematic presented in Fig. 1.2

In the *first* contribution (**Perception, Chapter 2**), we formulate the 3D human pose reconstruction from multiview images as a sparse optimization problem, which generates solutions that are certifiably globally optimal while enforcing realistic constraints on motion. In the *second* contribution (**Monitoring, Chapter 3**), published in [92] and [93], we compute "Stability Basins" to characterize stability during the sit-to-stand motion and experimentally demonstrate that these objects predict instability. In the *third* contribution (**Manipulation, Chapter 4**), published in [94] (which extends our work on quadrotors [95]), we efficiently compose zonotope-based reachable sets of robotic manipulators to plan guaranteed collision-free trajectories in real time. In the *fourth* contribution (**Fall Prevention, Chapter 5**), building on our preprint [96], we apply this reachable

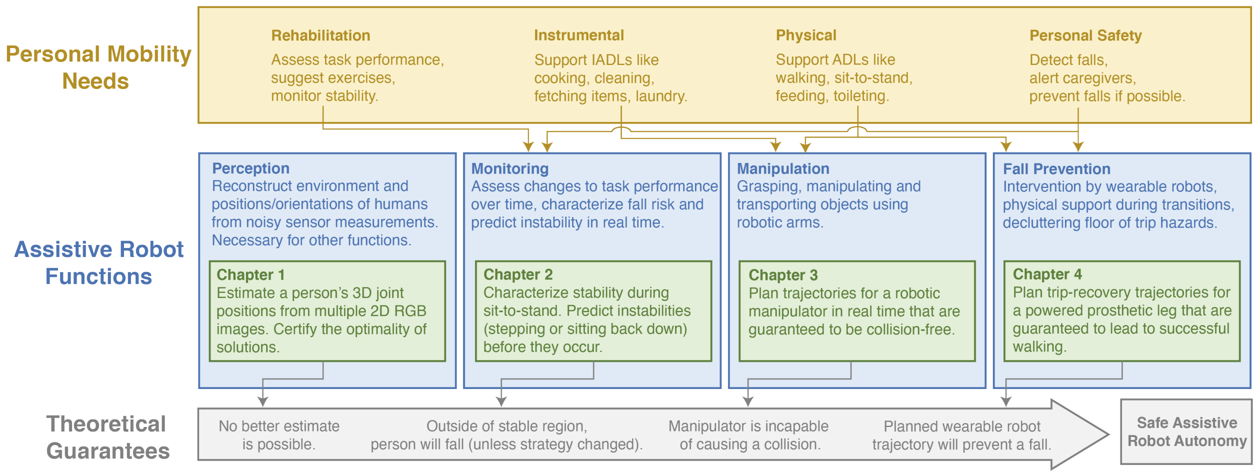

Personal Mobility Needs
Rehabilitation
Assess task performance, suggest exercises, monitor stability.
Instrumental
Support IADLs like cooking, cleaning, fetching items, laundry.
Physical
Support ADLs like walking, sit-to-stand, feeding, toileting.
Personal Safety
Detect falls, alert caregivers, prevent falls if possible.
Assistive Robot Functions
Perception
Reconstruct environment and positions/orientations of humans from noisy sensor measurements. Necessary for other functions.
Chapter 1
Estimate a person's 3D joint positions from multiple 2D RGB images. Certify the optimality of solutions.
Monitoring
Assess changes to task performance over time, characterize fall risk and predict instability in real time.
Chapter 2
Characterize stability during sit-to-stand. Predict instabilities (stepping or sitting back down) before they occur.
Manipulation
Grasping, manipulating and transporting objects using robotic arms.
Chapter 3
Plan trajectories for a robotic manipulator in real time that are guaranteed to be collision-free.
Fall Prevention
Intervention by wearable robots, physical support during transitions, decluttering floor of trip hazards.
Chapter 4
Plan trip-recovery trajectories for a powered prosthetic leg that are guaranteed to lead to successful walking.
Theoretical Guarantees
No better estimate is possible.
Outside of stable region, person will fall (unless strategy changed).
Manipulator is incapable of causing a collision.
Planned wearable robot trajectory will prevent a fall.
Safe Assistive Robot Autonomy

Figure 1.2: Dissertation summary. This dissertation develops a set of tools to enable safe autonomy in assistive robots for supporting personal mobility needs. Each chapter of the dissertation presents a different tool and provides theoretical guarantees about their correct performance. These theoretical guarantees appear in Sec. 2.3.4 (optimal perception), Thm. 3.6.2 (instability during motion), Thm. 4.4.15 (collision-free manipulation), and Thm. 5.5.6 (fall prevention).

set trajectory planning framework to the challenge of online trip recovery in robotic prosthetic legs. Though not included in this document, we note that our pose estimation contribution is inspired by our work on automating assessment of rehabilitation following ACL reconstruction surgery (detailed in our preprint [97]), and that our work with reachability analysis began with the computation of reachable sets under parametric uncertainty, published in [98].

These contributions are presented in greater detail below:

1.4.1 Perception of 3D pose from 2D images

In this contribution, we formulate an optimization-based framework for reconstructing 3D human pose from 2D estimates in multiple camera views. This algorithm is based on the Sparse Bounded Sums-of-Squares (SBSOS) hierarchy, which has been applied in other settings for finding globally-optimal solutions that minimize a sum of measurement residuals [99, 100]. When compared to current state-of-the-art methods [4], our algorithm matches or exceeds their accuracy without the need for 3D training data, is less computationally expensive, and can incorporate intuitive constraints on motion. Additionally, the algorithm provides an easily-checkable condition to determine whether its solutions are globally optimal. Experiments show global optimality is achieved at the first stage of the SBSOS hierarchy over 99% of the time.

1.4.2 Monitoring Instability during Sit-to-Stand

In this contribution, we compute and experimentally validate Stability Basins, which characterize the region inside which a human is stable during a dynamic task. Stability Basins can be used for monitoring a person's stability and to predict when a change in control strategy is necessary to prevent a fall in real time. In this work we focus on the sit-to-stand (STS) task, which is a dynamic motion necessary for maintaining independence and quality of life [101]. Given an individualized dynamic model, Stability Basins [92] represent the set of model states that lead to successful completion of the task. Here, we show how to construct Stability Basins from kinematic observations of STS, and demonstrate the accuracy of Stability Basin predictions with an 11-person experiment in which subjects were pulled by motor-driven cables as they stood from a chair. The methods and results of this study are presented in [93], which builds on our previous work [92]. Stability Basins are found to be accurate for a STS strategy preferred by older persons, and show promise for longitudinal studies that examine their relationship to fall risk.

1.4.3 Planning Collision-free Trajectories for Manipulators

Reachability-based Trajectory Design (RTD) is a receding-horizon trajectory planner that can guarantee the safety of its plans; however, it was previously limited to low dimensional 2D planning models [89]. In this contribution, we extend RTD to planning for high-dimensional robotic manipulators in 3D. This method is called ARMTD (Autonomous Reachability-based Manipulator Trajectory Design). ARMTD leverages low-dimensional reachable sets of each joint angle to efficiently compose the reachable set of the full arm in workspace. This allows for each joint of the arm to be controlled independently, while accounting for the effect each joint angle has on the positions of the arm's links. Online, these reachable sets are intersected with obstacles to identify strict collision-avoidance constraints. ARMTD then solves an optimization problem subject to these constraints, which returns a trajectory that is guaranteed to be collision-free. The planner is parallelized with each step implemented on a GPU to achieve real-time performance. ARMTD is demonstrated in simulation as well as on hardware with a Fetch Mobile Manipulator robot, where it never causes a collision. This work is detailed in [94] and builds off our previous work on 3D quadrotor flight [95].

1.4.4 Planning Trip Recoveries for Prostheses to Prevent Falls

In this contribution, we adapt the ARMTD framework to design trip-recovery trajectories for a powered lower-limb prosthesis. This method is called TRIP-RTD (Trip Recovery in Prostheses via Reachabililty-based Trajectory Design) and is designed to prevent falls in transfemoral amputees. TRIP-RTD utilizes a planning model composed of intact hip and prosthetic subsystems. We consider the case where a person trips when their prosthetic leg is in the swing phase (e.g. if the prosthetic toe unexpectedly catches on an obstacle). When a trip occurs, TRIP-RTD combines a predicted reachable set of the hip subsystem with a parameterized reachable set of the prosthetic subsystem. These reachable sets are used to define safety constraints, where safety is enforced using a "target set" of desired swing foot positions, and a clearance constraint that prevents the prosthesis from unexpectedly catching on the ground. An online optimization then generates a trajectory for the prosthetic leg that is safe for any trajectory in the set of predicted human motion. In a simulation experiment using data of two non-amputees responding to trips, TRIP-RTD finds a safe trip-recovery trajectory in 74.11% of trials. We are currently preparing an experiment involving tether-trip perturbations to further test the method and begin implementing TRIP-RTD on hardware.

1.5 Preliminaries

1.5.1 Notation

For brevity, we introduce notation here that will be followed throughout this document. Variables, points, and functions are lowercase; sets and matrices are uppercase. The n-dimensional real numbers are $\mathbb{R}^n$, natural numbers are $\mathbb{N}$, the unit circle is $\mathbb{S}^1$, and the set of 3×3 rotation matrices is $\mathsf{SO}(3)$. Vectors are either $[x_1, \cdots, x_n]^\top$ or $(x_1, \cdots, x_n)^\top$. For vectors $v^1, v^2, ...v^p \in \mathbb{R}^n$, $[v^1, v^2, ..., v^p] \in \mathbb{R}^{n \times p}$ is a matrix formed by concatenating the vectors. The positive real line is $\mathbb{R}_+ = [0, +\infty)$. Let $U, V \subset \mathbb{R}^n$. For a point $p \in U$, $\{p\} \subset U$ is the set containing p. The power set of U is $\mathcal{P}(U)$. The Minkowski sum is $U \oplus V = \{u + v \mid u \in U,\ v \in V\}$. Let $U \times V$ denote the Cartesian product of sets U and V. For a matrix $A \in \mathbb{R}^{n \times n}$, $AU = \{Au \mid u \in U\}$. For a state x, its first (resp. second) time derivative is $\dot{x}$ (resp. $\ddot{x}$). The interval $[0, T]$ is the set $\{t \mid 0 \leq t \leq T\}$. Superscripts on points index elements of a set. Subscripts index dimensions or provide contextual information.

1.5.2 Concepts

Several concepts will appear repeatedly throughout this document, which we detail here. Frequently, we let $x : [0, T] \to X$ denote a *state trajectory* of a system or model, where X is the *state space*. The state trajectory maps time to states of the system according to some system dynamics, and we say that $x(t)$ is the state of the system at time t. Similarly, we often let $q : [0, T] \to Q$ denote a *configuration trajectory* of a system or model, where Q is the *configuration space*, and say

$q(t)$ is the configuration at time t. Generally, we use the term "configuration" to refer to positions of multi-link robots, and the more general concept of a robot's "state" when including velocities. In Chaps. 4 and 5 we utilize *parameterized configuration trajectories* for planning. This will employ the notation $q(\cdot; k) : [0, T] \to Q$, where the *trajectory parameter* $k \in K$ is used to select a particular configuration trajectory.

Next, we introduce *zonotopes*. A zonotope Z is a polytope in $\mathbb{R}^n$ that is closed under linear maps and Minkowski sums [102], and is parameterized by its center $c \in \mathbb{R}^n$ and generators $g^1, ...g^p \in \mathbb{R}^n$. A zonotope describes the set of points that can be written as the center c plus a linear combination of the generators, where the coefficient β^i on each generator must be between -1 and 1:

$$Z = \left\{ y \in \mathbb{R}^n \;\middle|\; y = c + \sum_{i=1}^{p} \beta^i g^i,\ -1 \leq \beta^i \leq 1 \right\} \tag{1.1}$$

For convenience, we may concatenate the generators into an $n \times p$ *generator matrix* G, and the coefficients into a *coefficient vector* β:

$$G = \left[g^1, g^2, ..., g^p\right] \text{ and } \beta = [\beta^1, \beta^2, \cdots, \beta^p]^\top. \tag{1.2}$$

We can then rewrite (1.1) as

$$Z = \{y \in \mathbb{R}^n \mid y = c + G\beta,\ -1 \leq \beta \leq 1\}, \tag{1.3}$$

where $\geq$ and $\leq$ are applied elementwise. For brevity, we will henceforth assume that $\beta \in [-1, 1]$, and will write the constraints explicitly when this is not true. Zonotopes are a subclass of polytopes amenable to reachable set computation [103]. An illustration of a zonotope is given in Fig. 1.3.

Future sections make use of *reachability analysis* to compute *reachable sets* represented by zonotopes. Briefly, reachability analysis studies the flow of sets of states of a system under the system's dynamics [103].

To explain, we consider a linear time-invariant system $\dot{x}(t) = Ax(t)$, where $A \in \mathbb{R}^{n \times n}$ and $x(t) \in \mathbb{R}^n$. Trajectories of this system take the form $x(t) = e^{At}x_0$ where x_0 is an initial state. Given some intial set of states $X_0 \subset \mathbb{R}^n$ and these dynamics, the reachable set of the system at time t is the set of states $\Phi_t(X_0) = \{y \in \mathbb{R}^n \mid y = e^{At}x_0,\ x_0 \in X_0\}$. In other words, the reachable set at time t is the set of valid trajectories that originate from X_0 evaluated at time t. The reachable set of states over the interval $t \in [0, T]$ is then $\bigcup_{t \in [0,T]} \Phi_t(X_0)$.

Throughout this dissertation, we will compute over-approximations of reachable sets. This is useful for ensuring safety properties. For example, if a set of states does not cause a collision

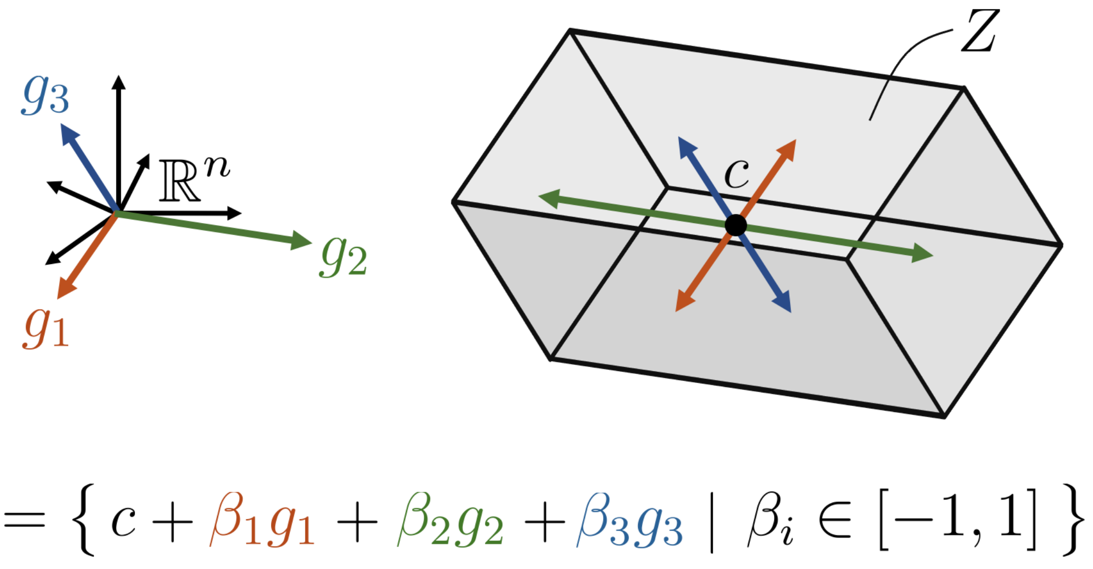

g_3
$\mathbb{R}^n$
g_2
g_1
Z
c
$Z = \{ c + \beta_1 g_1 + \beta_2 g_2 + \beta_3 g_3 \mid \beta_i \in [-1, 1] \}$

Figure 1.3: A zonotope Z is a polytope in $\mathbb{R}^n$ described by a center and generator vectors. This figure depicts a zonotope with three generators. Starting at the center, the zonotope is the set of points which may be written as a linear combination of the generator vectors, where each generator may be multiplied by a coefficient in $[-1, 1]$.

with an obstacle, then any state contained within that set certainly does not either. We use the open-source reachability toolbox called CORA for computing over-approximations of reachable sets [102, 104]. CORA efficiently handles difficulties associated with representing sets and their flow under a system's dynamics. CORA represents reachable sets as a collection of zonotopes at each of a finite collection of *time steps*, which are subintervals of some time interval $[0, T]$ of length Δt. With an abuse of notation, we let t act as an index, so that $Z^{(t)}$ is the zonotope overapproximating the reachable set over the time step that contains t. Briefly, CORA works by linearizing the dynamics at each time step about the center of $Z^{(t)}$, and obtaining $Z^{(t+\Delta t)}$ by multiplying $Z^{(t)}$ by an overapproximation of the matrix exponential over that time step. It then expands $Z^{(t+\Delta t)}$ to account for the effects of inputs and linearization error.

We note that an initial set X_0 may only contain a single state $X_0 = \{x_0\}$. In this case, one may still obtain reachable sets (as opposed to a single state trajectory) if the system's dynamics are uncertain or parameterized by a set of trajectory parameters K.

The set of all states that can be attained by the system starting from some initial set is referred to as the *forward reachable set*. However, in Chap. 3, we are interested in state trajectories which end in some final set. In practice, this *backward reachable set* can be computed by treating the final set as an initial set and reversing the direction of the system's dynamics.

CHAPTER 2

Perceiving Globally-optimal 3D Human Pose from Multi-view 2D Images

2.1 Introduction

In this chapter, we consider the challenge of **perceiving** 3D positions of a human's joints from multiple 2D RGB cameras simultaneously collecting video. We focus on obtaining **globally-optimal** solutions to this problem, which means that with our choice of cost function and constraints, no better solution exists.

Human pose estimation is a fundamental computer vision problem with important applications across many fields. Accurate perception of human pose is especially important for enabling safe autonomy in assistive robot applications. For autonomous navigation and manipulation, assistive robots need to precisely locate a person's body to avoid injurious collisions. For wearable robots, offboard sensors in a person's home may provide important contextual information for their control, such as identifying that a person is about to climb a set of stairs. Pose estimation can also be used for monitoring a person's risk of falling in real time, and alerting wearable robots if a trip or slip is sensed. In addition to safety concerns, pose estimation can allow assistive robots to more seamlessly interact with a person. While fetching and presenting an object to a person, a manipulator could use estimates of a person's arm and hand position to facilitate a handoff to the person. Accurate estimates of a person's head and mouth are necessary for the task of supporting feeding for persons with limited upper-limb mobility. Pose estimation can be used to communicate with robots via gesture recognition, and for interpreting cues for predicting a person's intent [105]. Finally, long term monitoring of human kinematics in the home made available by pose estimation algorithms may provide new avenues of study in the field of movement science [106] and understanding biomechanical health.

2.1.1 Related Work

Modern human pose estimation methods can be split into monocular (single view) versus multi-view methods, depending on the number of cameras used for inference. Recently, much research has focused on estimating 3D human pose using monocular methods [107]. This is challenging because of the depth ambiguity inherent to a single camera view. However, recent approaches using neural networks have shown great promise even in this challenging scenario [108].

Multi-view methods are important for applications such as robotics or biomechanics, when more than one camera or sensor is likely available to use. For example, many robots will have a stereo camera, LIDAR, or other depth sensor [109]. The multi-view case is potentially simpler because the depth of a point can be resolved by triangulating multiple sensors. In the canonical case where 2D estimates are perfect projections of a 3D position, this may be done via a singular value decomposition [110]. However, a more robust approach is necessary when noise is present in the 2D estimates [111].

Recent multi-view approaches are dominated by neural networks. Generally, these approaches use a two-stage framework [112]. In the first stage, 2D estimates of joint locations are collected for each individual camera using a neural network. In the second stage, the 2D estimates are aggregated to generate a 3D reconstruction, also usually by a neural network. These approaches have been shown to produce excellent accuracy on certain datasets, such as the popular Human 3.6m dataset [4, 113]. An illustration of this two-stage approach is provided in Fig. 2.1.

However, these datasets utilize validation images that closely match the neural networks' training data. In the Human 3.6m dataset, the camera views and actions performed by the subjects are identical. All that changes between the training and validation data are the subjects themselves [1]. It remains unclear how well these approaches generalize to real-world scenarios because of this reliance on extensive training data. Furthermore, within these approaches it is often unclear how to incorporate intuitive constraints on human motion. Most human joints are connected by rigid links of a fixed length, but this link length constraint is non-convex [114], which makes it difficult to encode exactly within a neural network. Additionally, inferences about 3D pose are usually generated for a single video frame (rather than a sequence of frames) because incorporating temporal information is difficult [115].

In contrast, we propose an optimization-based approach for 3D pose reconstruction. This still follows a two-stage approach; in the 1st stage we use the same neural networks to generate 2D estimates for each camera. However, we replace the 2nd stage neural network (which reconstructs 3D pose from 2D estimates) with an optimization problem. This has several advantages. First, the optimization problem requires no training data. Though training data is readily available for 2D pose estimation (because it is easy to accurately annotate 2D images) [116, 117], training data for 3D pose estimation is much more limited because it typically requires a motion capture system for

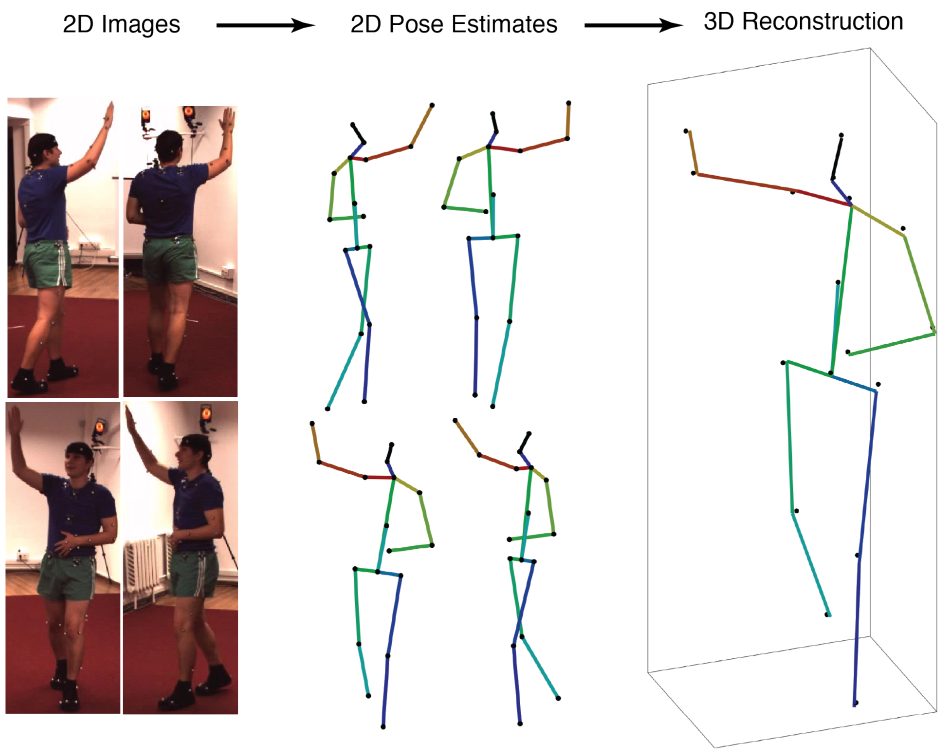


Figure 2.1: An example of the two-stage pose estimation framework applied to a test frame from the Human 3.6m [1] dataset. First, 2D pose estimates are collected from multiple RGB images showing different views of the same subject. Then, given the extrinsic camera parameters which describe how the cameras are oriented with respect to each other, the multiple views are used to produce an accurate 3D pose estimate.

estimating the ground truth 3D joint positions [1]. Second, the optimization problem can employ strict constraints on link length. Third, the optimization problem may be more computationally efficient in terms of memory and time required for inference. Fourth and perhaps most importantly, solutions to this optimization can be certified as globally optimal with an easily checkable matrix rank condition, meaning that a better 3D pose estimate does not exist.

The framework we propose is based on the Sparse Bounded Degree Sums-of-Squares (SBSOS) hierarchy of convex relaxations [118]. Given a polynomial optimization problem, the SBSOS hierarchy gives a sequence of convex relaxations of the original problem that converge to the global solution. The globally-optimal solution of the original problem is found at the first step of the hierarchy if certain conditions are satisfied. This means that for problems with certain properties, global minima can be found quickly and efficiently by exploiting the sparsity of the cost and constraint functions. Recently, SBSOS has been used to find solutions to robot estimation problems such as SLAM [99] and rotation averaging [100]. Additionally, similar hierarchies of convex relaxations have shown promise for use in outlier-robust estimation [119].

2.1.2 Contributions

Our contributions are threefold. First, we formulate the 3D pose reconstruction problem given 2D pose estimates as an optimization problem, with strict constraints on link length. Second, we use SBSOS to define this problem, where certificates of global optimality are produced at the first stage of the relaxation hierarchy over 99% of the time. Third, we perform a number of experiments on the Human 3.6m dataset with multiple 2D pose detectors, and compare our method to the current state of the art. We show our method matches or exceeds the accuracy of the current state of the art, with attractive theoretical guarantees and a shorter inference time.

2.2 Formulating the Optimization Problem

This section describes our method for reconstructing 3D human pose, starting with 2D estimates in individual camera frames and ending with the SBSOS formulation of the problem. The following notation will be adopted throughout this chapter and is explained in Fig. 2.2.

2.2.1 Preliminaries

Let $J = \{1, 2, ..., n_J\}$ index a set of n_J *joints*. Let $K = \{1, 2, ..., n_K\}$ index a set of n_K *cameras*. We use $x^j \in \mathbb{R}^3$ to represent our decision variables, which will be the 3D positions of joint $j \in J$

in some global coordinate frame:

$$x^j = \begin{bmatrix} x_1^j \\ x_2^j \\ x_3^j \end{bmatrix} \tag{2.1}$$

Let $\boldsymbol{x} = (x^1, x^2, ..., x^{n_J})^\top$ be the concatenation of all decision variables, where $\boldsymbol{x} \in \mathbb{R}^{3n_J}$.

We use $y^{jk} \in \mathbb{R}^2$ to represent a 2D estimate (i.e. pixel location) of joint j from camera $k \in K$. Also, let $w^{jk} \in [0, 1]$ represent a *confidence* in the estimate y^{jk}. Let $L \subset J \times J$ index pairs of *linked joints*, i.e.

$$(p, q) \in L \implies ||x^p - x^q||_2 = l^{pq} \tag{2.2}$$

where l^{pq} is the length of the link connecting joint p to joint q. See Fig. 2.2 for a visualization of these variables. In total, say that there are n_L links connecting the n_J joints. We assume the following about link lengths:

Assumption 2.2.1. *The set of linked joints* L *and estimates of the link lengths* l^{pq} *connecting any two joints* p *and* q *are given.*

In Sec. 2.4, we show that we can use either ground truth link length data or estimate the link lengths via unconstrained 3D pose estimation.

Let $c^k \in \mathbb{R}^3$ be the position of the k-th camera in the global coordinate frame. Let $R^k : \mathbb{R}^3 \to \mathbb{R}^3$ be a function that transforms the decision variables in world coordinates to each camera-centric coordinate system (whose origin coincides with c^k). Finally, let $P^k : \mathbb{R}^3 \to \mathbb{R}^2$ be functions that project 3D points in each camera's coordinate frame onto camera pixel coordinates (e.g. using homogeneous coordinates).

We assume the following about these parameters:

Assumption 2.2.2. *Estimates of the cameras' extrinsic and intrinsic parameters* R^k *and* P^k *are given.*

These parameters may either be measured *a priori* or estimated online, which may be necessary if the cameras are moving.

2.2.2 Generating 2D human pose estimates

We use 2D estimates y^{jk} of each joint j in each camera frame k, as well as a confidence w^{jk} for each estimate. However, our method is indifferent to how these estimates are produced. In Sec. 2.4, we show that our method works with multiple different 2D pose estimation algorithms. We consider y^{jk} and w^{jk} to be the output of some black box function which takes k RGB images as input.

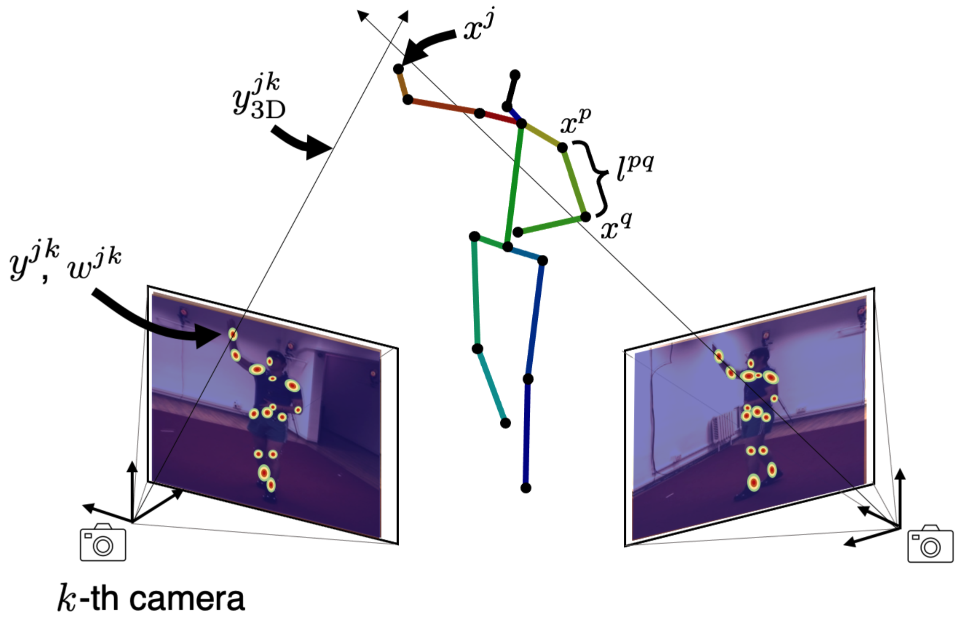


Figure 2.2: This figure explains notation and concepts found in this chapter. 2D pose estimates are generated in each camera frame for each joint $j \in J$. The estimated 2D pixel location of joint j in the k^{th} camera frame is given by y^{jk}, with confidence w^{jk}. These estimates are "unprojected" to form a ray y^{jk}_{3D} of 3D points that may have produced the estimate. The decision variables x^j are chosen to minimize a squared measurement residual representing the distance of x^j from each of these rays. Link length constraints are enforced between any two linked joints; in this figure, joints p and q are connected by a link of length l_{pq}.

2.2.3 The 3D reconstruction optimization problem

We begin by formulating the (unconstrained) version of the 3D reconstruction optimization problem. We consider a single frame of a video sequence. We seek to minimize a weighted sum of squared *measurement residuals*, finding the 3D positions of each joint that best explain the estimates produced by each camera. However, the decision variables are in 3D global coordinates, and the estimates are in 2D pixel coordinates.

We use the position of the camera center c^k and the estimate y^{jk} to draw a ray in 3D space which represents 3D positions that may have produced y^{jk}. First, we "unproject" the estimate y^{jk} using homogeneous coordinates and by inverting the projection function, similar to the "volumetric" method in [4]. This point is then rotated to the global coordinate system. Letting

$$a = c^k \tag{2.3}$$

and

$$b = R^{k^{-1}}(P^{k^{-1}}(\begin{bmatrix} y_1^{jk} \\ y_2^{jk} \end{bmatrix})), \tag{2.4}$$

the set of points that lie on this ray is given by

$$y_{\text{3D}}^{jk} = \{x \in \mathbb{R}^3 \,|\, x = a + bu, u \in [0, +\infty)\}. \tag{2.5}$$

This ray is depicted in Fig. 2.2. We have made use of the camera's extrinsic and intrinsic parameters R^k and P^k, which are assumed to be given as in Assum. 2.2.2.

We want to choose the x^j that minimizes the distance from any point on the ray y_{3D}^{jk}. Note that for a given x^j, this distance is minimized by a point $v \in y_{\text{3D}}^{jk}$ such that $x^j - v$ is perpendicular to the vector $b - a$. After some rearranging, the minimum distance $r(x^j, y_{\text{3D}}^{jk})$ between x^j and y_{3D}^{jk} is given by

$$r(x^j, y_{\text{3D}}^{jk}) = \frac{||(b-a) \times (a - x^j)||_2}{||b-a||_2} \tag{2.6}$$

where $\times$ denotes the cross product and $|| \cdot ||_2$ denotes the Euclidean norm [120]. We refer to the value $r(x^j, y_{\text{3D}}^{jk})$ as the **measurement residual**, which represents the error incurred by a choice of decision variable x^j with respect to the "unprojected" measurement y_{3D}^{jk}. For brevity, we simply write this $r(x^j, y^{jk})$, where the "unprojection" step is inferred.

Finally, the cost function we seek to minimize is the sum of the weighted squares of these measurement residuals:

$$\min_{\boldsymbol{x} \in \mathbb{R}^{3n_J}} \quad \textstyle\sum_{j \in J, k \in K} w^{jk} r^2(x^j, y^{jk}) \tag{2.7}$$

where w^{jk} is the confidence associated with the estimate y^{jk}.

For brevity, we remove the subscripts from the summation:

$$\min_{\boldsymbol{x} \in \mathbb{R}^{3n_J}} \quad \sum w^{jk} r^2(x^j, y^{jk}). \tag{2.8}$$

Essentially, instead of computing the measurement residual in the image plane, we compute the residual with respect to a ray passing through the joint estimate in 3D space. One drawback of this "unprojection" is that when the camera is very far from the person it is estimating, there can be a large cost associated with errors. In reality, we expect that a camera being far away should produce inaccurate pose estimates. These measurements therefore should incur small penalties within the cost function, where the desired weighting can be captured by adjusting the confidences w^{jk}. We note that in practice, if a camera does not observe a joint (and produces no y^{jk} and w^{jk}), we leave the corresponding term out of the summation.

2.2.4 Introducing constraints

Most of the 3D positions of the joints to be estimated are constrained by rigid link lengths that join them. Strictly incorporating these constraints within the optimization can improve the quality of the solution by only considering physically feasible joint positions. Recall that L indexes pairs of linked joints. We assume that the distance between pairs of linked joints is known, or has been estimated, as in Assum. 2.2.1. We add the constraint that the distance between pairs of linked joints is equal to this fixed length:

$$\begin{array}{cl} \min\limits_{\boldsymbol{x} \in \mathbb{R}^{3n_J}} & \sum w^{jk} r^2(x^j, y^{jk}) \\ \text{s.t} & {l^{pq}}^2 - ||x^p - x^q||_2^2 = 0 \quad \forall (p,q) \in L \end{array} \tag{2.9}$$

where we have squared the lengths in the constraint to avoid taking a square root (while maintaining the same feasible set). This constraint is depicted in Fig. 2.2.

2.2.5 Reformulation for SBSOS

Before introducing the SBSOS hierarchy for solving (2.9), we reformulate the equality constraints using inequality constraints.

$$
\begin{array}{lll}
\min\limits_{\boldsymbol{x}\in\mathbb{R}^{3n_J}} & \sum w^{jk} r^2(x^j, y^{jk}) & \\
\text{s.t} & 0 \leq {l^{pq}}^2 - ||x^p - x^q||_2^2 \leq 1 & \forall (p,q) \in L \\
& 0 \leq 1 + {l^{pq}}^2 - ||x^p - x^q||_2^2 \leq 1 & \forall (p,q) \in L.
\end{array}
\tag{2.10}
$$

Notice that these two inequality constraints can only be satisfied simultaneously when $l^{pq} - ||x^p - x^q||_2 = 0$, as desired.

Finally, to further simplify notation, let

$$f^{jk}(x^j) = w^{jk} r^2(x^j, y^{jk}), \tag{2.11}$$

$$g_{ub}^{pq}(x^p, x^q) = {l^{pq}}^2 - ||x^p - x^q||_2^2 \tag{2.12}$$

$$g_{lb}^{pq}(x^p, x^q) = 1 + {l^{pq}}^2 - ||x^p - x^q||_2^2 \tag{2.13}$$

Then, we have:

$$
\begin{array}{lll}
\min\limits_{\boldsymbol{x}\in\mathbb{R}^{3n_J}} & \sum f^{jk}(x^j) & \\
\text{s.t} & 0 \leq g_{ub}^{pq}(x^p, x^q) \leq 1 & \forall (p,q) \in L \\
& 0 \leq g_{lb}^{pq}(x^p, x^q) \leq 1 & \forall (p,q) \in L.
\end{array}
\tag{2.14}
$$

2.3 Sparse Bounded Degree Sums of Squares Programming

Next, we describe how we use SBSOS to formulate and solve the optimization problem (2.14).

First, we note that each squared measurement residual $r^2(x^j, y^{jk})$ is a degree-2 polynomial in $\mathbb{R}[x_1^j, x_2^j, x_3^j]$. To see this, note that

$$r^2(x^j, y^{jk}) = \frac{1}{||b-a||_2^2}((b-a)\times(a-x^j))^\top((b-a)\times(a-x^j)), \tag{2.15}$$

and that each element of the vector $((b-a)\times(a-x^j))$ is a linear combination of the elements of x^j. Because the cost function is a weighted sum of these squared residuals, it is also a degree-2 polynomial in $\mathbb{R}[\boldsymbol{x}]$. Furthermore, each of the length constraints presented in (2.14) are also degree-2 polynomials. The cost and constraints in (2.14) are quadratic polynomials of the decision variables, so (2.14) is a **quadratically constrained quadratic program** [121].

Unfortunately, the problem is non-convex because of the link length constraints [114]. To understand why, consider a person's arm, which we model as three joints connected by two links. Imagine a sphere centered at the person's shoulder with a radius equal to the length of their upper arm, and a second sphere centered at the person's wrist with a radius equal to the length of their lower arm. The position of the person's elbow has to be the correct length from both the shoulder and the wrist, and therefore must lie on the surface of each sphere. This means that the feasible set of the elbow position is defined by the intersection of these spheres (assuming they do intersect). Unless the two spheres meet at a single point, this intersection is a circle in $\mathbb{R}^3$, which is not convex.

2.3.1 Overview

Because (2.14) is a non-convex quadratically constrained quadratic program, it is known to be NP-hard and difficult to solve in general [121]. Fortunately, hierarchies of convex relaxations exist which converge to the globally-optimal solution of the problem as the degree of the relaxation goes to infinity (and often earlier) [87, 88]. A variety of such hierarchies exist, but this work focuses on the Bounded Degree Sums-of-Squares (BSOS) [122] hierarchy, and its sparse implementation Sparse-BSOS (SBSOS) [118].

BSOS works by optimizing over sum-of-squares (SOS) polynomials of a *bounded degree*, where the degree is chosen by the user ahead of time. SOS polynomials are guaranteed to be positive, and BSOS utilizes SOS polynomials to ensure that positivity constraints are satisfied for feasible solutions. BSOS finds the best such polynomials to minimize the cost function at the optimal solution.

Each of these programs can be implemented as a semidefinite program (SDP) because one can represent SOS polynomials as semidefinite matrices [87]. SDPs are convex, and efficient solvers exist to solve them [123]. However, the number of variables needed to represent these SOS polynomials quickly increases depending on the stage of the hierarchy and the user's choice of bounded degree. The resulting increase in runtime and memory usage of the SDP can make the problem computationally intractable. To address these computational challenges, we take advantage of the sparsity inherent in our pose estimation problem via the SBSOS hierarchy. To introduce the SBSOS formulation, it is easier to first formulate the BSOS program before bringing in the structured sparsity leveraged by SBSOS.

2.3.2 BSOS Programming

Consider a general unconstrained polynomial optimization problem, with decision variables $\boldsymbol{x} \in \mathbb{R}^N$ and cost $f \in \mathbb{R}[\boldsymbol{x}]$. The problem $\inf_{\boldsymbol{x} \in \mathbb{R}^N} f(\boldsymbol{x})$ is equivalent to

$$t^* = \sup_{t \in \mathbb{R}} \{t \mid f(\boldsymbol{x}) - t \geq 0, \, \forall \, \boldsymbol{x}\}. \tag{2.16}$$

Next, consider the **semialgebraic set** $K \subset \mathbb{R}^n$, defined by polynomials $g_j \in \mathbb{R}[\boldsymbol{x}]$, $j = 1, ..., M$:

$$K = \{\boldsymbol{x} \in \mathbb{R}^N \,|\, g_j(x) \geq 0,\, j = 1, ..., M\} \tag{2.17}$$

We introduce the constraint that $\boldsymbol{x} \in K$, yielding

$$\sup_{t \in \mathbb{R}} \{t \,|\, f(\boldsymbol{x}) - t \geq 0,\, \forall\, \boldsymbol{x} \in K\}. \tag{2.18}$$

Here, one would have to enforce that $f(\boldsymbol{x}) - t \geq 0\, \forall\, \boldsymbol{x} \in K$, while simultaneously optimizing over t so that t was as large as possible at the optimal solution. In practice, one method for enforcing the constraint $f(\boldsymbol{x}) - t \geq 0\, \forall\, \boldsymbol{x} \in K$ is to introduce a polynomial $h \in \mathbb{R}[\boldsymbol{x}]$ as a decision variable. Let us suppose for now that we could enforce $h(\boldsymbol{x}) \geq 0\, \forall\, \boldsymbol{x} \in K$. Then, we could optimize

$$\sup_{t \in \mathbb{R}, h \in \mathbb{R}[\boldsymbol{x}]} \{t \,|\, f(\boldsymbol{x}) - t - h(\boldsymbol{x}) \geq 0,\, \forall\, \boldsymbol{x}\}. \tag{2.19}$$

Here, we have replaced the semialgebraic set constraint by the polynomial h, where we have that $f(\boldsymbol{x}) - t \geq h(\boldsymbol{x}) \geq 0\, \forall\, \boldsymbol{x} \in K$. By jointly optimizing over t and h, we enforce the semialgebraic set constraint while simultaneously trying to make t as large as possible at the optimal solution.

The BSOS (bounded degree sums-of-squares) hierarchy's name reflects two key ideas. The first idea is to utilize sums-of-squares polynomials to approximate the constraint $f(\boldsymbol{x}) - t - h(\boldsymbol{x}) \geq 0$ by requiring $f(\boldsymbol{x}) - t - h(\boldsymbol{x}) \in \Sigma[\boldsymbol{x}]$, where $\Sigma[\boldsymbol{x}] \subset \mathbb{R}[\boldsymbol{x}]$ is the ring of SOS polynomials in $\boldsymbol{x}$, which provides a sufficient condition for positivity. The second idea is to represent $h(\boldsymbol{x})$ using a polynomial of *bounded degree*, which is computationally attractive because it fixes the size of a semidefinite matrix used to represent the semialgebraic set constraint a priori.

BSOS requires a user to specify two parameters: $d \in \mathbb{N}$ and $k \in \mathbb{N}$. The parameter d bounds the degree of the polynomial used to represent h, and the parameter k bounds the degree of the SOS polynomials used to represent the constraint.

First, we discuss the parameter d. Let $\mathbb{N}^{2M}_d = \{(\alpha, \beta) \mid \alpha, \beta \in \mathbb{N}^M, |\alpha| + |\beta| \leq d\}$, where the absolute value denotes the sum of the elements of these vectors. Then, according to [122, Equation 4], the following polynomial is positive on K:

$$h_d(\boldsymbol{x}, \boldsymbol{\lambda}) := \sum_{(\alpha,\beta) \in \mathbb{N}^{2M}_d} \lambda_{\alpha\beta} \prod_{j=1}^{M} g_j(\boldsymbol{x})^{\alpha_j} (1 - g_j(\boldsymbol{x}))^{\beta_j} \tag{2.20}$$

where $\boldsymbol{\lambda} = (\lambda_{\alpha\beta})$ are nonnegative and we assume that $0 \leq g_j \leq 1$ on K, which can be achieved via scaling. Here, d is used to bound the maximum degree of each monomial in the sum, while the coefficients of the polynomial $h_d(\boldsymbol{x}, \boldsymbol{\lambda})$ are specified by $\lambda_{\alpha\beta}$. We will replace $h(\boldsymbol{x})$ in (2.19) with

$h_d(\boldsymbol{x}, \boldsymbol{\lambda})$, thereby generating a lower bound on the original problem.

Next, we discuss the parameter k. Recall that SOS polynomials are guaranteed to be positive everywhere, and that the space of SOS polynomials $\Sigma[\boldsymbol{x}] \subset \mathbb{R}[\boldsymbol{x}]$. The parameter k bounds the degree of SOS polynomials under consideration, where $\Sigma[\boldsymbol{x}]_k \subset \mathbb{R}[\boldsymbol{x}]_{2k}$ represents the space of SOS polynomials with degree of at most $2k$ (which are sums of polynomials with degree of at most k that have been squared).

Using these together, we can generate a hierarchy of convex relaxations of the original problem (2.19) that converges to the true solution for any k as $d \to \infty$ [122, Theorem 2(a)]:

$$q_d^k = \sup_{t, \boldsymbol{\lambda}} \{t | f(\boldsymbol{x}) - t - h_d(\boldsymbol{x}, \boldsymbol{\lambda}) \in \Sigma[\boldsymbol{x}]_k, \boldsymbol{\lambda} \geq 0\} \tag{2.21}$$

where $\geq$ is understood element-wise. Note that in this formulation, one optimizes over the specific polynomial $h_d(\boldsymbol{x}, \boldsymbol{\lambda})$ by changing its coefficients $\boldsymbol{\lambda}$, where $h_d(\boldsymbol{x}, \boldsymbol{\lambda})$ is guaranteed to be positive on K. So, the goal is to simultaneously make $h_d(\boldsymbol{x}, \boldsymbol{\lambda})$ as small as possible at the optimal solution, so that t can be as large as possible at the optimal solution. The problem can be represented as a semidefinite program (because SOS polynomials can be represented by semidefinite matrices), and is therefore convex.

Though q_d^k is guaranteed to converge as $d \to \infty$, convergence actually occurs at the first step of the relaxation for an important class of problems, i.e. q_1^1 gives the optimal t^* [122, Theorem 2(b)]. Unfortunately, the pose estimation problem we seek to solve (2.14) does not fall into this class. However, in practice q_1^1 often still gives the optimal solution to our problem, as discussed in Sec. 2.4.

Though BSOS provides bounds on the memory required to represent (2.21), the problem becomes computationally intractable when larger numbers of decision variables and constraints are desired, or for higher degree polynomial cost and constraint functions. A sparse version of BSOS was developed to address these issues, called Sparse-BSOS (SBSOS) [118], which we discuss next.

2.3.3 SBSOS Programming

SBSOS shares all the features of BSOS described in the last subsection, but takes advantage of the fact that many optimization problems exhibit structured *sparsity*. This means that although there may be many decision variables, the problem can be broken into smaller blocks, where the cost function and constraints may only depend on the variables within a block. Thus, the memory requirements needed to represent positivity constraints are reduced, and the problem may be solved much faster. To formalize this, SBSOS requires that the problem can be broken into $p \in \mathbb{N}$ blocks satisfying the **Running Intersection Property (RIP)**, which we state subsequently.

Given $A \subset \{1, ..., N\}$, let $\mathbb{R}[\boldsymbol{x}; A]$ denote the ring of polynomials in variables $\{x_i \mid i \in A\}$.

Definition 2.3.1. *Running Intersection Property (RIP): There exists $p \in \mathbb{N}$ and $A_\ell \subseteq \{1, ..., N\}$ and $B_\ell \subseteq \{1, ..., M\}$ for all $\ell \in \{1, ..., p\}$ such that:*

- *$f = \sum_{l=1}^{p} f^\ell$, for some $f^1, ..., f^p$ such that $f^\ell \in \mathbb{R}[\boldsymbol{x}, A_\ell]$ for each $\ell \in \{1, ...p\}$,*
- *$g_j \in \mathbb{R}[\boldsymbol{x}, A_\ell]$ for each $j \in B_\ell$ and $\ell \in \{1, ..., p\}$,*
- *$\cup_{l=1}^{p} A_\ell = \{1, ..., N\}$,*
- *$\cup_{l=1}^{p} B_\ell = \{1, ..., M\}$,*
- *for all $\ell = 1, ..., p-1$, there is an $s \leq \ell$ such that $\left(A_{\ell+1} \cap \cup_{r=1}^{\ell} A_r\right) \subseteq A_s$.*

Colloquially, the last condition of the RIP can be understood as the requirement that "when creating subsequent blocks, any variables that have appeared before must have all appeared in at least one preceding block".

With these blocks defined, we can begin to break the BSOS problem (2.21) into smaller pieces. Let $\mathbb{N}^\ell := \{(\alpha, \beta) \mid \alpha, \beta \in \mathbb{N}^M, \text{supp}(\alpha) \cup \text{supp}(\beta) \subseteq B_\ell\}$ where $\text{supp}(\alpha) := \{j \in \{1, ..., M\} \mid \alpha_j \neq 0\}$. Also, let $\mathbb{N}_d^\ell := \{(\alpha, \beta) \in \mathbb{N}^\ell \mid |\alpha| + |\beta| \ \leq d\}$. We create polynomials $h_d^\ell(\boldsymbol{x}, \boldsymbol{\lambda}) \in \mathbb{R}[\boldsymbol{x}; A_\ell]$ that only depend on the variables indexed by A_ℓ and are guaranteed to be positive on the subset of K associated with the constraints in B_ℓ:

$$h_d^\ell(\boldsymbol{x}, \boldsymbol{\lambda}^\ell) := \sum_{(\alpha,\beta) \in \mathbb{N}_d^l} \lambda_{\alpha\beta}^\ell \prod_{j=1}^{M} g_j(\boldsymbol{x})^{\alpha_j} (1 - g_j(\boldsymbol{x}))^{\beta_j} \tag{2.22}$$

where the vector of coefficients $\boldsymbol{\lambda}^\ell = (\lambda_{\alpha\beta}^\ell)$ is required to be nonnegative.

By creating p of these polynomials $h_d^\ell(\boldsymbol{x}, \boldsymbol{\lambda}^\ell), \ell \in \{1, ..., p\}$, we can reduce the size of the semidefinite set constraint required to enforce positivity on K. Similarly to the BSOS hierarchy, the SBSOS hierarchy generates convex relaxations of the original problem that converge to the true solution for any k as $d \to \infty$ [118, Theorem 2]:

$$\begin{aligned} q_d^k = \sup_{\substack{t, \boldsymbol{\lambda}^1, ... \boldsymbol{\lambda}^p \\ f^1, ..., f^p}} \{t \mid & f^\ell(\boldsymbol{x}) - t - h_d^\ell(\boldsymbol{x}, \boldsymbol{\lambda}^\ell) \in \Sigma[\boldsymbol{x}; A_\ell]_k, \ell = 1, ..., p \\ & f(\boldsymbol{x}) - t = \sum_{l=1}^{p} f^\ell(\boldsymbol{x}), \boldsymbol{\lambda}^\ell \geq 0, f^\ell \in \mathbb{R}[\boldsymbol{x}; A_\ell]_{d_{\max}}\} \end{aligned} \tag{2.23}$$

where $d_{\max}$ is defined on page 7 of [118].

2.3.4 Certificates of global optimality

Similarly to BSOS, under certain assumptions, the SBSOS hierarchy converges to the globally-optimal solution at the first stage of the hierarchy [118, Theorem 3]. A recent paper published by my colleagues [99] purported to prove that a problem similar to our pose estimation problem (2.14) satisfied these assumptions; however, their proof was found to be in error [124]. We remain confident that a proof of convergence at the *second* stage of the hierarchy exists (i.e. $d = 2, k = 2$), and continue to work on a formal proof of this.

Despite the lack of a proof of convergence, one can verify whether or not a solution produced by SBSOS is globally optimal. This requires checking the ranks of a set of matrices [118, Lemma 4], where one matrix is associated with each block of decision variables A_ℓ. If the matrices are all rank 1 (i.e. there is only one linearly-independent column/row), then the solution is certified to be globally optimal. We show in Sec. 2.4 that a globally-optimal solution to (2.14) is obtained the over 99% of the time at the first stage of the hierarchy (i.e. $d = 1, k = 1$). This attractive property of the framework provides the theoretical guarantee of optimality described in Sec. 1.4 and the dissertation summary presented in Fig. 1.2.

2.3.5 Implementation

We use SBSOS to formulate the pose estimation problem (2.14). First, we must ensure our problem satisfies RIP (Def. 2.3.1). We do so by choosing A_ℓ as the decision variables associated with any pair of linked joints and B_ℓ as the link length constraints that join them. This results in $p = n_L$ blocks of at most 6 decision variables each (3 for each joint), which is a significant reduction compared to the $3n_J$ total decision variables. By leveraging sparsity and enforcing positivity on these smaller blocks of variables, the computational burden of the SDPs needed to solve these problems is drastically reduced. As a consequence, (2.14) may be quickly solved, as demonstrated in Sec. 2.4. We note that *no initial guess* is required for this optimization. The optimality check in Sec. 2.3.4 is very computationally efficient, and for our problem only requires an eigendecomposition of a set of n_J matrices that are each 7×7.

All experiments are performed in MATLAB [125] using modified versions of the code provided in [118] and [99] to formulate the optimization problems. The SDP solver within the optimization library MOSEK [123] is used to solve each problem. Note that in Sec. 2.4, we generate 2D pose estimates using algorithms described in [4] and [116]. The outputs of these algorithms are stored, then loaded in MATLAB for use in optimization.

Lastly, we note that SOS programming becomes ill-conditioned when optimizing over a large domain. To counteract this, we scale each problem so that the decision variables $\boldsymbol{x}$ are required to lie within the unit box.

Actions (absolute positions, **all** cameras)	Dir.	Disc.	Eat	Greet	Phone	Pose	Purch.	Sit	SitD.	Smoke	Photo	Wait	Walk	WalkD.	WalkT.	Avg
Triangulation (w/o conf.)	22.1	24.8	24.1	21.4	28.0	20.8	23.3	33.6	37.0	28.1	23.9	21.6	22.4	24.6	23.2	25.3
Iskakov et al. [4], algebraic	18.1	20.0	17.6	17.0	18.9	17.4	19.2	21.9	23.2	19.5	19.3	18.0	18.3	20.5	17.9	19.2
Iskakov et al. [4], volumetric (softmax)	16.9	**18.1**	16.6	**16.0**	17.9	16.5	18.5	**19.6**	**20.1**	18.2	**17.1**	**16.8**	**17.2**	**19.0**	**16.6**	**17.7**
Ours, LT detector, w/o length constraints	18.5	20.4	18.0	17.6	19.6	17.8	20.2	22.7	23.5	20.1	19.9	18.4	19.3	21.3	18.0	19.7
Ours, LT detector, w/ length constraints	**16.8**	18.8	**15.8**	16.4	**17.4**	**16.4**	**18.4**	19.7	20.2	**17.9**	17.6	17.1	17.8	**19.0**	16.9	**17.7**

Table 2.1: Comparison of our method to the current state of the art [4] on the Human 3.6m dataset. Results are presented as the absolute MPJPE (in mm) from ground truth. The validation set has been filtered according to the protocol done by Iskakov et al. [4], which removed scenes with erroneous ground truth annotations. Minimum MPJPE in each column is bolded. The same LT 2D pose detector is used for our method as for Iskakov et al.'s methods to generate the 2D pose estimates, and all camera views are included for inference. Furthermore, we include results for an unweighted naive triangulation method, which serves as a baseline for comparison.

2.4 Experiments

All experiments are performed on the Human 3.6m dataset [1]. We perform experiments using 2D pose estimates generated via two distinct methods. In the first experiment, we generate 2D pose estimates and confidences using the "Learnable Triangulation" PyTorch implementation created by Iskakov et al. [4]. In the second experiment, these estimates and confidences are produced using OpenPose with the default settings [116]. While the "Learnable Triangulation" 2D pose detector is trained on the Human 3.6m dataset, OpenPose is not. Therefore, the "Learnable Triangulation" experiments show how our method compares to the current state of the art, while the OpenPose experiments demonstrate that our method is indifferent to how the 2D pose estimates are generated, and can therefore generalize to other datasets or 2D detectors.

Human 3.6m is a popular 3D pose estimation dataset, which includes four camera views imaging simultaneously at 50Hz and motion capture markers for determining ground truth joint positions [1]. The dataset is split into training and validation datasets, with 5 actors used for training and 2 actors used for validation. The actors perform a sequence of actions, such as "walking a dog" or "giving directions". The actors are the only individuals in the scene, and are instructed not to step out of a certain area to keep their whole body in the frame of each camera. Furthermore, there are no props or objects in the room which could occlude the actors.

Typically, results for this dataset are reported as Mean Per-Joint Position Error (MPJPE) in mm. This is the average L2 distance between the estimated and ground truth joint positions across joints, image frames, and validation subjects. In many single-view 3D pose estimation papers, the MPJPE results are reported relative to the pelvis keypoint [112], as the authors are less concerned with accurately estimating the absolute positions of the keypoints. In multi-view applications, estimating absolute positions is more important, hence the results here are given for absolute MPJPE.

For a fair comparison, we follow the validation protocol done by the current state-of-the-art methods proposed by Iskakov et al. [4]. Certain actions for one validation subject were removed

Method	Avg. 3D Inference Time (s)
Triangulation (w/o conf.)	0.002
Iskakov et al. [4], algebraic	0.002^*
Iskakov et al. [4], volumetric (softmax)	0.302^*
Ours, LT detector, w/o length constraints	0.018
Ours, LT detector, w/ length constraints	0.027

Table 2.2: Comparison of inference time between our method and the current state of the art [4] on the Human 3.6m dataset when using all available cameras. Inference time for our method is the time to solve the optimization problem using SBSOS with $k = 1$, $d = 1$. For the methods used by Iskakov et al. [4], inference time has been reported for the full pipeline [5, Tab. 4], which we use to infer the time for the 3D reconstruction alone. The same LT 2D pose detector is used in all instances. *Here, we assume that the "algebraic" method takes the same amount of 3D inference time as the nearly identical unweighted "triangulation" approach, and use this to arrive at the time for the "volumetric" approach.

due to an erroneous offset in ground truth keypoint positions. Furthermore, another validation action has data from only three of the four cameras. Every fifth frame is used for evaluation.

2.4.1 Comparison to current state of the art

We utilize the pretrained 2D backbone found in Iskakov et al. to generate 2D keypoints estimates and confidences. We refer to this as the **LT** 2D pose detector (for "Learnable Triangulation") in the tables in this section. This backbone is trained on data from the COCO and MPII datasets [4, 126, 117] before further training on Human 3.6m itself. We emphasize that our 3D pose estimation optimization problem requires *no training data*, though in this experiment, the 2D detector we use is trained on the Human 3.6m dataset. For the case where length constraints are included in our optimization, we use the ground truth data to find these link lengths. However, in the next subsection 2.4.2 we show that our method can use link lengths estimated from the images instead.

Results for a direct comparison to the state-of-the-art methods are displayed in Tab. 2.1. Our method with length constraints performs comparably to the "volumetric" method, having the same average error across subjects and actions. We emphasize that the "volumetric" 3D pose estimation method is trained on data from Human 3.6m, while our method only requires a subject's link lengths. While achieving comparable accuracy, our method takes less than $1/10$th the inference time to reconstruct the 3D pose, reported in Tab. 2.2.

Our method without length constraints performs slightly poorer than the "algebraic" method on most actions. This is expected because the confidences w^{jk} we employ are generated to work specifically for the "algebraic" method. These results demonstrate that with views from all four cameras, the 3D pose reconstruction can be performed very accurately without the need for constraints when using this 2D pose detector.

Human 3.6m provides 360 degree views of subjects from the four camera angles, which may

Actions (absolute positions, **front** cameras)	Dir.	Disc.	Eat	Greet	Phone	Pose	Purch.	Sit	SitD.	Smoke	Photo	Wait	Walk	WalkD.	WalkT.	Avg
Iskakov et al. [4], algebraic	24.3	28.0	22.4	23.8	24.7	24.0	24.0	27.1	30.8	25.7	24.6	24.8	25.1	28.5	25.4	25.6
Iskakov et al. [4], volumetric (softmax)	28.1	28.0	23.2	25.8	26.2	23.2	22.2	26.5	29.9	28.6	30.1	24.8	28.5	30.2	29.6	27.0
Ours, LT detector, w/o length constraints	24.3	27.7	22.5	23.7	24.7	23.9	24.0	26.9	30.6	25.6	24.6	24.8	25.2	28.6	25.4	25.5
Ours, LT detector, w/ length constraints	**21.8**	**24.8**	**19.2**	**21.6**	**21.7**	**21.9**	**22.1**	**23.7**	**26.2**	**23.2**	**21.5**	**23.0**	**23.3**	**26.2**	**22.8**	**22.9**
Actions (absolute positions, **back** cameras)	Dir.	Disc.	Eat	Greet	Phone	Pose	Purch.	Sit	SitD.	Smoke	Photo	Wait	Walk	WalkD.	WalkT.	Avg
Iskakov et al. [4], algebraic	28.3	31.0	37.8	28.1	39.0	25.0	31.4	52.8	56.3	38.0	33.1	27.7	25.1	30.4	25.8	34.0
Iskakov et al. [4], volumetric (softmax)	33.1	38.4	35.7	30.9	42.5	25.3	36.8	48.8	68.1	39.5	57.7	29.7	24.5	33.4	25.1	38.0
Ours, LT detector, w/o length constraints	28.3	31.0	31.6	28.0	37.3	24.6	31.7	42.8	54.1	36.5	33.4	27.7	25.2	30.6	25.8	32.6
Ours, LT detector, w/ length constraints	**24.9**	**27.7**	**27.0**	**25.6**	**34.9**	**22.8**	**28.7**	**36.4**	**47.0**	**33.0**	**30.3**	**25.4**	**23.4**	**27.1**	**23.4**	**29.2**

Table 2.3: Comparison of our method to the current state of the art [4] on the Human 3.6m dataset when using subsets of cameras. Results are presented as the absolute MPJPE (in mm) from ground truth. The validation set has been filtered according to the protocol done by Iskakov et al. [4], which removed scenes with erroneous ground truth annotations. Furthermore, the scene "Directions-2" has been removed from these results due to the availability of data from only 3 cameras. Minimum MPJPE in each column is bolded. The same LT 2D pose detector is used for our method as for Iskakov et al.'s methods.

not be realistic for robotics applications. To make the dataset more challenging, we ran the same experiment, but removing two camera views at a time. In the first test, we only used the two cameras at the "front" of the room, and in the second test, we used the two cameras at the "back" of the room. This makes the 3D pose estimation problem much more difficult, and can be thought of as stereo vision with an extra-wide baseline. The results of this experiment are given in Tab. 2.3. Here, our method with length constraints outperforms the state of the art methods across all scenes, showing its utility in more difficult scenarios and suggesting overfitting in the "volumetric" method, as it performs worse than the "algebraic" method.

Finally, as described in Sec. 2.3.4, when using all available cameras our method can verify the global optimality of its 3D pose estimates in a vast majority of test frames. Averaging across actions, this happens 99.29% of the time at the first stage of the SBSOS hierarchy ($d = 1$ and $k = 1$) when length constraints are included, as reported in Tab. 2.4. However, 0.24% of the time, a solution is returned after the second stage ($d = 2$ and $k = 2$) of the hierarchy which is not verified as globally optimal. This may be due to the presence of multiple globally-optimal solutions, or numerical issues within the optimization.

2.4.2 Experiments using OpenPose

The experiments in the previous section (Sec. 2.4.1) utilized a 2D pose detector which generates state-of-the-art results on the Human 3.6m dataset. One reason for its high accuracy is because it is trained directly on Human 3.6m. Though Human 3.6m's training and testing data subsets contain different actors, the camera placement, lighting conditions, and actions performed by the actors are identical. Furthermore, the "Learnable Triangulation" backbone *jointly* infers 2D pose estimates and confidences in each RGB image, instead of considering each RGB image separately. This is

Actions ($\%$ certified solutions, **all** cameras)	Dir.	Disc.	Eat	Greet	Phone	Pose	Purch.	Sit	SitD.	Smoke	Photo	Wait	Walk	WalkD.	WalkT.	Avg
Ours, LT detector, w/o length constraints, $d=1, k=1$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Ours, LT detector, w/o length constraints, $d=2, k=2$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Ours, LT detector, w/ length constraints, $d=1, k=1$	99.88	99.79	99.95	99.60	99.97	100.00	98.65	99.60	98.60	97.84	99.05	98.73	100.00	98.66	100.00	99.29
Ours, LT detector, w/ length constraints, $d=2, k=2$	99.94	99.35	99.95	100.00	100.00	100.00	99.38	99.85	99.55	98.70	99.93	99.94	100.00	99.72	100.00	99.76

Table 2.4: Percentage of test frames for each action where our method finds a solution that is verified as globally optimal when using all available cameras. See Sec. 2.3.2 for an explanation of the parameters d and k, which specify the convex relaxation in (2.23). Also see Sec. 2.3.4 to understand how we certify the global optimality of solutions. For our method without length constraints, a globally-optimal result is always given at the 1st stage ($d = 1, k = 1$) of the SBSOS hierarchy. For our method with length constraints, a solution that is not verified as globally optimal is produced 0.24% of the time after the 2nd stage ($d = 2, k = 2$).

key to their contribution of "learning the triangulation" of a set of cameras; however, if the cameras are reconfigured, it is unclear how to adapt the method without requiring additional training data. In their paper, the authors do adapt their method to the CMU panoptic dataset [127], but require their networks to be retrained. A heavy reliance on 3D ground-truth training data provided by motion capture limits the generalizability of neural-network-based 3D pose estimation algorithms.

In contrast, training data for 2D pose estimation is readily available, because 2D annotations of images are easier to generate. Ideally, a 3D pose estimation method would only require 2D pose estimates (and known/estimated camera calibration parameters), with no requirement for 3D training data or joint inference on the set of 2D images. This would maximize their ability to operate in arbitrary scenarios, which is necessary for robotics applications where cameras may be constantly rearranged. To demonstrate the generalizability of our method, we ran experiments using 2D pose estimates generated via OpenPose [116]. We refer to this as the **OP** detector. OpenPose is trained on the COCO [117] and MPII [126] datasets. *No additional training on the Human3.6m dataset is performed.* Furthermore, OpenPose makes inferences on each image *individually*, in contrast to the "Learnable Triangulation" 2D pose detector, which makes inferences *jointly*.

In Sec. 2.4.1, we enforced link length constraints based on the ground truth link length data. Here, we estimate the link lengths by first running the unconstrained version of our optimization problem, then taking the average distance between the estimated 3D positions of linked joints. We note that these link length estimates do not necessarily match the ground truth link lengths. With the link lengths estimated, the only additional data required by our 3D pose algorithm for these experiments are the camera extrinsic and intrinsic parameters (as in Assum. 2.2.2). We also note that OpenPose produces 2D estimates for a slightly different set of keypoints than is found in the Human3.6m dataset. We use the closest possible keypoints for which ground truth data exists, (i.e. the ground truth nose position is compared to the OpenPose's forehead estimate), and skip keypoints (like the mid spine) for which OpenPose never produces an estimate.

While the "Learnable Triangulation" 2D backbone takes 2 seconds to jointly infer 2D esti-

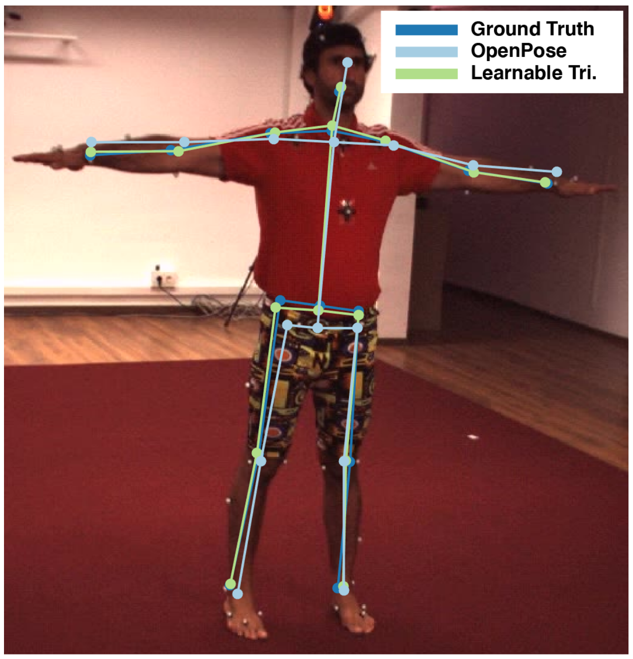


Figure 2.3: A comparison of 2D keypoint estimates for the "Learnable Triangulation" detector and OpenPose versus the ground truth. Because OpenPose is not trained on Human3.6m, its 2D estimates can differ significantly from the ground truth annotations, contributing to a drop in accuracy. For example, the OpenPose's hip position estimates (light blue) are consistently lower and more narrow than the ground truth (dark blue) or "Learnable Triangulation" (light green) estimates.

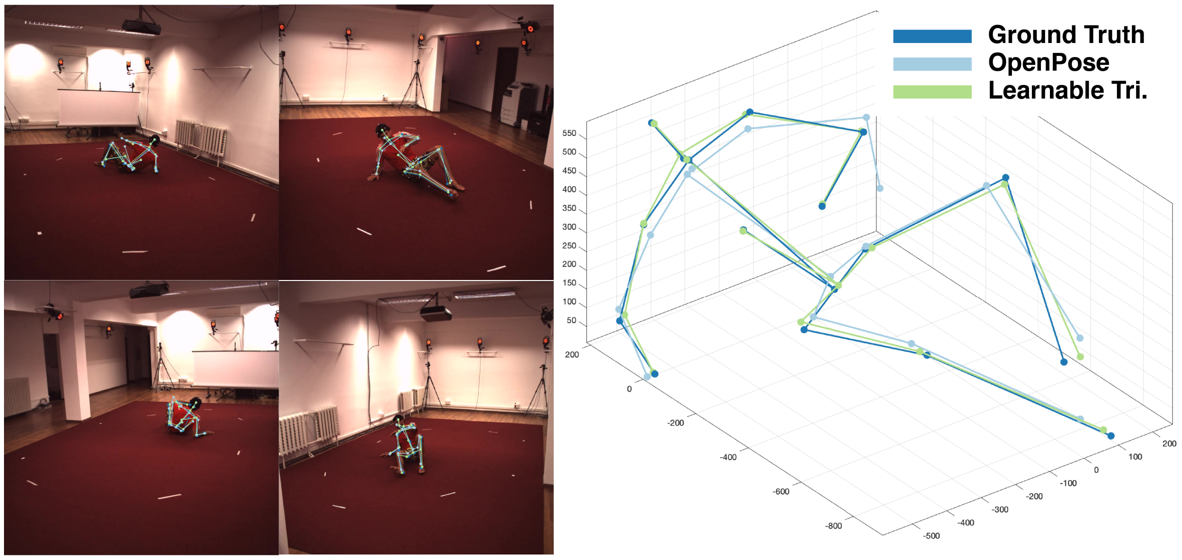


Figure 2.4: A comparison of 3D pose estimates generated via our method with length constraints when using the LT and OP detectors. The 3D estimate formed using OpenPose is relatively accurate. However, it misplaces the 3D position of the left arm because of poor estimates from the back cameras (leftmost two images on the left hand side).

mates in four camera frames [5], OpenPose runs much faster, with speeds of 22 frames per second reported on a machine with an Nvidia GTX 1080 Ti graphics card [116]. This increase in speed comes at the cost of accuracy, as reported in Tab. 2.5. On average, our method paired with the OpenPose detector yields a MPJPE of 52.6mm, versus an average of 17.7mm when paired with the "Learnable Triangulation" backbone. However, much of OpenPose's MPJPE comes from a mismatch between the kinematic models used to train OpenPose and to create the Human3.6m ground truth data. For example, OpenPose consistently estimates the hip joint positions to be narrower than is given in the Human3.6m ground truth data, as shown in Fig. 2.3. A comparison between the 3D pose estimates using both 2D detectors for our method with length constraints is presented in Fig. 2.4.

Experiments using OpenPose on the "front" and "back" camera subsets are given in Tab. 2.7. Here, the inclusion of length constraints yields a larger increase in accuracy compared to when all cameras are made available for inference. In these more challenging scenarios, the link length constraints contribute relatively more to overall estimation because it is less straightforward to infer an appropriate 3D joint position.

Finally, we find that our method with length constraints still returns a certifably-optimal solution 95.88% of the time at the first stage ($d = 1, k = 1$) of the SBSOS hierarchy when averaged across actions, displayed in Tab. 2.6. Surprisingly, our method without length constraints sometimes returns a solution which can not be certified as globally optimal. Upon closer inspection, the test

Actions (absolute positions, **all** cameras)	Dir.	Disc.	Eat	Greet	Phone	Pose	Purch.	Sit	SitD.	Smoke	Photo	Wait	Walk	WalkD.	WalkT.	Avg
Ours, LT detector, w/o length constraints	18.5	20.4	18.0	17.6	19.6	17.8	20.2	22.7	23.5	20.1	19.9	18.4	19.3	21.3	18.0	19.7
Ours, LT detector, w/ length constraints	**16.8**	**18.8**	**15.8**	**16.4**	**17.4**	**16.4**	**18.4**	**19.7**	**20.2**	**17.9**	**17.6**	**17.1**	**17.8**	**19.0**	**16.9**	**17.7**
Ours, OP detector, w/o length constraints	51.1	52.8	52.0	50.1	56.1	49.4	54.8	71.5	65.1	56.7	54.5	50.9	51.0	54.3	49.8	54.7
Ours, OP detector, w/ length constraints	49.9	51.7	47.2	48.1	55.1	47.7	53.3	68.2	61.9	54.1	52.5	49.6	49.3	52.9	47.4	52.6

Table 2.5: Comparison of our method using two different 2D pose detectors on the Human 3.6m dataset and all available cameras. Results are presented as the absolute MPJPE (in mm) from ground truth. The validation set has been filtered according to the protocol done by Iskakov et al. [4], which removed scenes with erroneous ground truth annotations. Minimum MPJPE in each column is bolded.

Actions ($\%$ certified solutions, **all** cameras)	Dir.	Disc.	Eat	Greet	Phone	Pose	Purch.	Sit	SitD.	Smoke	Photo	Wait	Walk	WalkD.	WalkT.	Avg
Ours, OP detector, w/o length constraints, $d = 1, k = 1$	100.00	100.00	99.95	100.00	100.00	100.00	93.89	99.16	97.03	100.00	99.86	99.10	100.00	100.00	100.00	99.27
Ours, OP detector, w/o length constraints, $d = 2, k = 2$	100.00	100.00	99.95	100.00	100.00	100.00	93.89	99.16	97.03	100.00	99.86	99.10	100.00	100.00	100.00	99.27
Ours, OP detector, w/ length constraints, $d = 1, k = 1$	98.16	96.80	98.98	97.43	95.40	99.70	92.75	91.67	90.98	96.69	89.09	95.74	99.39	97.67	97.79	95.88
Ours, OP detector, w/ length constraints, $d = 2, k = 2$	99.05	97.76	99.49	98.31	96.65	99.70	93.27	93.10	93.20	97.59	91.48	97.07	99.65	98.73	98.55	96.91

Table 2.6: Percentage of test frames where our method finds a solution which is verified as globally optimal when using the OpenPose detector on all available cameras. See Sec. 2.3.2 for an explanation of the parameters d and k, which specify the convex relaxation in (2.23). Also see Sec. 2.3.4 to understand how we certify the global optimality of solutions. For our method without length constraints, a globally-optimal result is given at the 1st stage ($d = 1, k = 1$) of the SBSOS hierarchy 99.27% of the time on average. In all cases where a result was not certified as globally optimal, at least one joint had only one 2D estimate, meaning multiple globally-optimal solutions exist. For our method with length constraints, solutions are verified as globally 95.88% of the time after the first stage ($d = 1, k = 1$) when averaged across actions.

frames where solutions could not be certified had only one 2D pose estimate for at least one joint. In these instances, multiple globally-optimal solutions exist, as the depth ambiguity of a particular joint can not be resolved from one estimate alone.

2.5 Conclusion

We presented a novel optimization-based algorithm for 3D human pose reconstruction that achieves state of the art performance on the Human 3.6m dataset, while requiring no training data. Our algorithm outperforms the state-of-the-art methods on more challenging scenarios created by removing camera views. The algorithm utilizes SBSOS to solve an optimization problem for estimating the 3D pose, which has several advantages. The method provides attractive theoretical guarantees for determining whether an estimate is a global optimizer, and is capable of quickly and efficiently finding such solutions. The optimization program can incorporate rigid link length constraints, and can potentially account for temporal continuity of motion across image frames. The fact that the method requires no 3D training data supports the conjecture that it can generalize to real-world scenarios, where annotated 3D pose data may be difficult to collect. This is further

Actions (absolute positions, **front** cameras)	Dir.	Disc.	Eat	Greet	Phone	Pose	Purch.	Sit	SitD.	Smoke	Photo	Wait	Walk	WalkD.	WalkT.	Avg
Ours, LT detector, w/o length constraints	24.3	27.7	22.5	23.7	24.7	23.9	24.0	26.9	30.6	25.6	24.6	24.8	25.2	28.6	25.4	25.5
Ours, LT detector, w/ length constraints	**21.8**	**24.8**	**19.2**	**21.6**	**21.7**	**21.9**	**22.1**	**23.7**	**26.2**	**23.2**	**21.5**	**23.0**	**23.3**	**26.2**	**22.8**	**22.9**
Ours, OP detector, w/o length constraints	63.2	66.8	64.1	65.7	64.4	61.5	59.4	62.8	79.3	67.8	59.8	64.3	82.1	80.5	76.7	67.9
Ours, OP detector, w/ length constraints	59.3	61.1	56.6	57.8	57.6	54.9	57.1	60.1	72.5	60.9	57.5	56.7	63.6	67.4	61.5	60.3
Actions (absolute positions, **back** cameras)	Dir.	Disc.	Eat	Greet	Phone	Pose	Purch.	Sit	SitD.	Smoke	Photo	Wait	Walk	WalkD.	WalkT.	Avg
Ours, LT detector, w/o length constraints	28.3	31.0	31.6	28.0	37.3	24.6	31.7	42.8	54.1	36.5	33.4	27.7	25.2	30.6	25.8	32.6
Ours, LT detector, w/ length constraints	**24.9**	**27.7**	**27.0**	**25.6**	**34.9**	**22.8**	**28.7**	**36.4**	**47.0**	**33.0**	**30.3**	**25.4**	**23.4**	**27.1**	**23.4**	**29.2**
Ours, OP detector, w/o length constraints	144.3	99.2	107.2	96.3	110.4	89.5	116.7	158.0	146.1	114.9	106.6	95.9	83.8	89.5	74.8	108.9
Ours, OP detector, w/ length constraints	214.7	72.2	76.4	67.1	89.7	60.4	81.3	123.4	111.9	87.9	81.5	70.9	62.8	73.1	60.6	88.9

Table 2.7: Comparison of our method using two different 2D pose detectors on the Human 3.6m dataset and subsets of cameras. Results are presented as the absolute MPJPE (in mm) from ground truth. The validation set has been filtered according to the protocol done by Iskakov et al. [4], which removed scenes with erroneous ground truth annotations. Furthermore, the scene "Directions-2" has been removed from these results due to the availability of data from only 3 cameras. Minimum MPJPE in each column is bolded.

demonstrated by testing using OpenPose to generate 2D pose estimates, which is not trained on the Human3.6m dataset.

Some major limitations of the method are that it so far has only been shown for the single-person case. However, if the 2D pose detection algorithms can reliably track a person, our method naturally extends to the multi-person case. Additionally, the method may not be robust to severe outliers in 2D pose detections. Future work will address this by incorporating robust cost functions within the optimization program.

One desirable aspect of our method is that it is indifferent to how the pose measurements are generated. Here, we use estimates from 2D RGB cameras, but one can imagine that data from other sensors (such as an IMU or 3D estimates from a LIDAR) could be incorporated within the optimization cost function. Future research will aim to optimally incorporate data streams from several sensors when estimating 3D pose.

The 3D pose estimation method presented in this chapter can be used as the first step in pipelines for safe autonomy for assistive robots. Because assistive robots rely on accurate perception for safety critical tasks, it is essential that these perception algorithms operate correctly. Our algorithm provides an easily-checkable condition for verifying the global optimality of its estimates. Therefore, assistive robots can be confident in the accuracy of the estimates produced by this method when providing personal mobility support.

CHAPTER 3

Monitoring Instability during Human Motion, Demonstrated on the Sit-to-Stand Task

Much of this chapter originally appeared in [93]:

> **Patrick D Holmes**, Shannon M. Danforth, Xiao-Yu Fu, Talia Y. Moore, and Ram Vasudevan. "Characterizing the limits of human stability during motion: perturbative experiment validates a model-based approach for the Sit-to-Stand task". *Royal Society Open Science*, 7:191410, (2020).

which extends our work in [92]:

> Victor Shia, Talia Y. Moore, **Patrick Holmes**, Ruzena Bajcsy, and Ram Vasudevan. "Stability basin estimates fall risk from observed kinematics, demonstrated on the sit-to-stand task". *Journal of Biomechanics* (2018).

Open source code for generating Stability Basins and visualizing the results presented in this chapter is available at `https://github.com/pdholmes/STS_SB`. *Kinematic sit-to-stand data from our cable-pull experiment may be found here:* `https://deepblue.lib.umich.edu/data/concern/data_sets/mw22v557c`.

3.1 Introduction

This chapter presents a tool for **monitoring instability** during human motion to enable assistive robots to mitigate the fall risk of older persons. Falls are the leading cause of injury in people over 75 years old, resulting in reduced quality of life, increased healthcare costs, and accident-related death [71, 81, 82]. If at-risk individuals can be identified prior to injury, the likelihood of falling can be significantly reduced through intervention such as physical therapy [128, 129]. Fall risk generally results from instability arising from neuromuscular deficiencies or external perturbations.

Clinical assessments for identifying individuals who would benefit from preventative care are currently limited to questionnaires [130, 131] and non-perturbative motor assessments [132, 133, 134]. Self-reported information often has low reliability [135, 136], and current clinical motor performance tests have low fall-prediction rates, especially for active older adults [137, 138, 139, 140]. For widespread use, stability assessments must combine predictive power with minimal experimental and computation time.

Assistive robots equipped with a suite of mobile sensors could assess stability over time by continuously monitoring a person's biomechanics [32]. For example, by repeatedly observing how a person completes a dynamic task such as sit-to-stand or walking, assistive robots could build personalized dynamic models of a person's motion. Changes to these models over time could be used to assess a person's risk of falling and provide recommendations to caregivers for rehabilitation exercises which mitigate fall risk. Additionally, wearable robots like powered prostheses and exoskeletons can provide support and stability during dynamic tasks like walking and sit-to-stand [141]. These robots have the potential to reduce the incidence of falls by detecting instabilities and responding appropriately. For example, during walking, a wearable robot can detect perturbations like slips in real time and choose an appropriate recovery action [30]. To determine when to implement a recovery action, the wearable robot should decide whether the person's current control strategy is sufficient to avoid a fall. In other words, the robot should predict when the person will become unstable.

Because older adults are more likely to fall while in motion [142], several studies suggest that quantifying dynamic stability may help identify biomechanical deficiencies associated with an increased risk of falling [143, 144, 145, 146]. A number of model-based methods have been developed to assess stability during walking [146]. Among these, variability measures [147] and the maximum Lyapunov exponent [148] ranked highest overall in validity. However, these metrics only characterize a subject's ability to recover from small perturbations, which a person may recover from without the need for robot intervention. To our knowledge, a verified technique for computing the maximum perturbation from which a subject can recover had not been created. Additionally, the best-performing model-based methods are limited to periodic motion.

Stability during non-rhythmic motion is of interest, especially because difficulty with aperiodic tasks such as sit-to-stand is strongly correlated with falls in older adults, and because these tasks are necessary for maintaining independence and quality of life [101]. The motion of a person's center of mass (COM) during sit-to-stand can be modelled as an inverted pendulum, which requires an appropriate amount of angular momentum at seatoff to successfully complete the task [149]. Drawing from this idea, metrics of stability for sit-to-stand are generally based on an individual's initial COM velocity or acceleration [150, 151, 152]. However, a stability metric that considers data only at the onset of movement disregards valuable information about the control strategy used by the individual. In fact, human control strategies for sit-to-stand lie on a spectrum ranging from Quasi-Static, in which little momentum is used and the body position is statically stable throughout the motion, to Momentum-Transfer, which is statically unstable and extensively uses momentum to achieve standing [153].

To account for the dynamic differences in sit-to-stand motions under distinct control strategies, our previous work introduced the Stability Basin, computed using individualized pendulum models of sit-to-stand with linear-quadratic regulator (LQR) controllers [92]. For this pendulum model, the Stability Basin at a given time is equal to the set of pendulum states that are able to successfully achieve standing under the specified controller. Though this Stability Basin under an LQR controller successfully distinguishes between less and more stable sit-to-stand strategies, its ability to accurately identify the set of states that can arrive at standing without failure of a given control strategy is unverified.

The aim of this research is to experimentally test whether Stability Basins can accurately estimate an individual's stability for a particular task. To do so, perturbations must be introduced in a way that causes the states of the individualized dynamic model to exit the stable region. By stability, we mean the set of body states through time from which a subject can successfully stand up without switching from their chosen control strategy to stepping or sitting. An accurate prediction of instability corresponds to an experimentally observed failure of the individual's control strategy. Here we use a perturbative sit-to-stand experiment with 11 subjects, described in Sec. 3.3, to validate stability predictions generated by the Stability Basin method.

3.2 Overview

This section describes our framework for computing and testing the accuracy of Stability Basins. First, we collect kinematic observations of a subject performing a perturbative sit-to-stand experiment (Sec. 3.3). Individualized biomechanical models of sit-to-stand are constructed for each subject (Sec. 3.4), and controller models are detailed in Sec. 3.5. Our proposed controller, which uses strategy-specific input bounds, reflects the distinct range of inputs required for each control strategy [154]. After computing the Stability Basins (Sec. 3.6) for each controller model, we test whether the individual and strategy-specific Stability Basins correctly predict when a subject steps or sits down in response to perturbation, and when they are successful (Sec. 3.6.4). Finally, we perform a comparison to a naive method for estimating stability (Sec. 3.6.3).

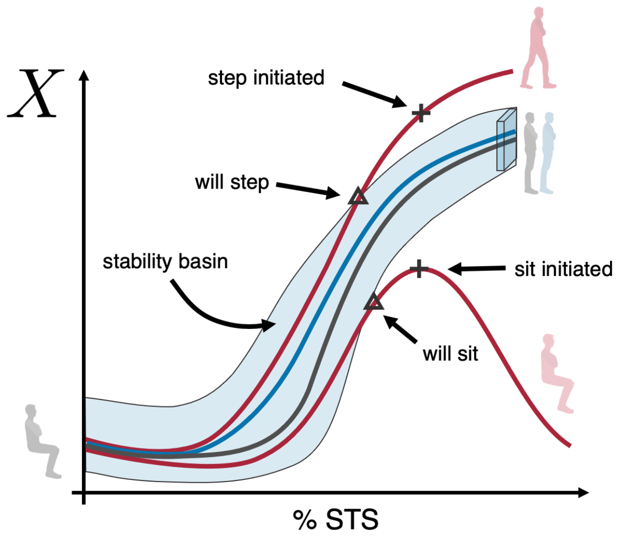


Figure 3.1: An illustrative overview of Stability Basins for sit-to-stand. The Stability Basin (shaded blue) represents the set of model states through time that will successfully arrive at a target set (light blue volume, right side) for a given individual and sit-to-stand strategy. Trajectories of the model are illustrated, with nominal in gray, successful perturbed in blue, and failures in red. Trajectories that exit the Stability Basin are predicted to lead to stepping or sitting.

3.3 Perturbative sit-to-stand experiment

This section describes the sit-to-stand experiment for testing Stability Basin predictions. The experimental protocol was approved by the University of Michigan Health Sciences and Behavioral Sciences Institutional Review Board, eResearch ID: HUM00020554.

Subjects began in a seated position on a stool with their arms crossed. The height of the stool was adjusted so that the subject's thighs were parallel to the ground. Subjects practiced standing up from the stool and were asked to find a comfortable foot position, which was then demarcated with a line of tape. Motion capture was used to collect kinematic data of subjects performing sit-to-stand. Subjects were instructed to perform three different sit-to-stand control strategies: their natural strategy, a Momentum-Transfer strategy, and a Quasi-Static strategy as done by Shia et al. [92]. Subjects watched a demonstration of each strategy, and then practised each strategy a minimum of 10 times prior to data collection. The following set of treatments were applied to each sit-to-stand strategy:

- Nominal trials: Subjects stood five times from a comfortable foot position using the specified control strategy.

- Foot-shift trials: Subjects varied their foot placement in 0.05-m (2-inch) increments in the anterior-posterior direction from their original position, which were demarcated by taped lines on the ground. Subjects stood up once at each increment, moving their feet backwards until their heels left the ground, and then forward until a strategy shift was observed. We noticed that subjects at extreme anterior foot positions attempted to stand up by lunging forward into a squatting position, and excluded these trials from our dataset because they represented a shift in strategy from the subject's nominal behavior.

- Cable pull perturbations: Subjects returned their feet to their original position. A cable system was attached to the subject's waist and connected to two high-torque motors. These motor driven cables applied impulses in the anterior-posterior direction to the subject as they rose. The cable pulls were restricted to the anterior-posterior direction, and were applied either forward or backwards with variable timing and force. Specifically, three peak force levels – low, medium, and high – were calibrated to each subject. The low force level was designed to rarely induce stepping or sitting during sit-to-stand, while the high force level was designed to induce stepping or sitting approximately half of the time. Six trials were taken at each force level for each sit-to-stand strategy, with three pulling forwards and three pulling backwards, in random order. For more details, see Sec. 3.3.1.

We collected 948 trials from 11 participants (three female and eight male; ages 18-32; height 1.70 ± 0.12 m; body mass 65.4 ± 10.2 kg), including 163 nominal, 194 foot-shifted, and 591 cable pull trials. Each subject gave their informed written consent, and had no physical or balance disorders which could affect their ability to perform sit-to-stand.

3.3.1 Generating cable pull perturbations

Two high-torque motors were attached to cable-pulley systems and placed in front of and behind the test platform. The cables were attached to a belt around the subject's waist, and the height of the pulleys were adjusted such that the cables were horizontal when the subject was standing. The pulleys were located approximately 3-4 feet in front of and behind the subjects. The cables were only attached for the cable pull condition, and detached for the nominal and foot-shift conditions.

A custom-written LabVIEW [155] program was used to control the motor torques. During the experiment, a low torque was constantly commanded in both motors to keep the cables from going slack. The torque level was balanced between the anterior and posterior cables to minimize force bias felt by the subject. In particular, both the front and back perturbation motors were commanded to maintain a baseline tensioning force of 19.6 N, so that the net force on the subject was approximately 0 N when no active perturbation was applied. Force sensors on the cables were synchronized with motion capture data, so that the timing of the perturbation application in relation to kinematic data was known. Perturbations were manually activated by the toggling of a handheld switch. The experiment operator applied perturbations to the subject at variable times following seatoff to gather a range of recovery responses over the course of sit-to-stand. Perturbations were generally applied at around 50% of the motion, shown in the Pert. Onset column of Table 3.1.

Perturbations were active over a 250 ms period, during which the applied force would ramp up to a specified peak, and then quickly ramp down to the baseline tensioning force, as shown in Fig. 3.2. The peak force to be applied was determined before the experiment began. Specifically, three peak force levels – low, medium, and high – were calibrated to each subject. The low force level was designed to rarely induce stepping or sitting during sit-to-stand, while the high force level was designed to induce stepping or sitting approximately half of the time. Typical values for the peak forces applied at each force level, normalized by each subject's body weight, are shown in Fig. 3.3. On average, peak forces were around 22% of body weight at the low force level, 32% at the medium force level, and 37% at the high force level.

Stepping and sitting are more formally defined in Sec. 3.6.5, and we report the number of steps and sits observed for each force level in Sec. 3.7. These force levels were roughly chosen based on the subject's height and weight, and adjusted manually during a pre-experiment test session. Six trials were taken at each force level for each sit-to-stand strategy, with three pulling forwards and three pulling backwards, in random order.

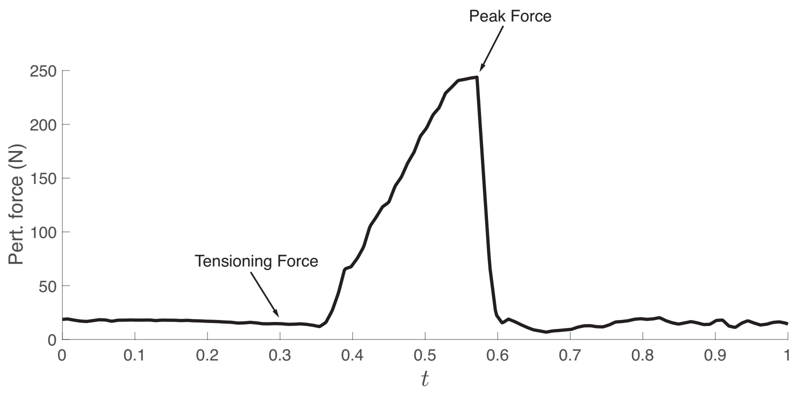


Figure 3.2: A typical perturbation time series is shown in this figure. This particular perturbation was a forwards perturbation at the medium force level. A baseline tensioning force of 19.6 N is commanded, until the start of the perturbation. The force quickly ramps up to a specified peak, then back down to the baseline tensioning force.

3.3.2 Kinematic observations

A 10-camera PhaseSpace motion capture system collected kinematic observations of 36 markers at 480 Hz. C-Motion's Visual3D biomechanics software was used to fit body segment models to each subject's data [156]. MATLAB was used for all subsequent analyses [125]. A 6th-order Butterworth filter with a cut-off frequency of 2 Hz was used to filter joint position trajectories.

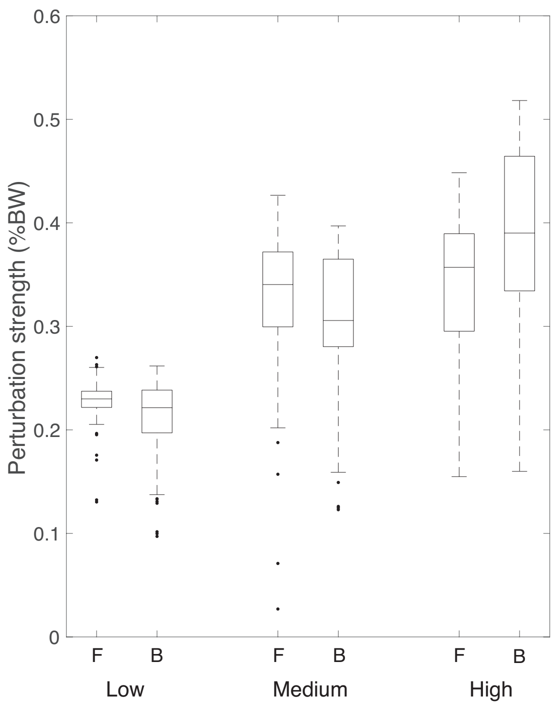


Figure 3.3: This figure shows typical peak perturbative force values (normalised by subject body weight) for the forwards (F) and backwards (B) perturbations applied at the low, medium, and high force levels. These peak forces were averaged across subjects and sit-to-stand strategies to produce this plot. For each force level, the central mark indicates the median peak force. The bottom and top edges of the box indicate the the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme forces not considered outliers, and the outliers are plotted individually.

We estimate the motion of the subject's center of mass (COM) from motion capture data. First, we use a 3-segment model to track the motion of the subject's shank, thigh, and head-arms-torso segments in the sagittal plane. The approximate COM of each segment is computed using anthropometric data [157], and combined to find the trajectory of the total body's COM position throughout sit-to-stand. Then, we obtain COM velocity and acceleration trajectories by numerically differentiating the COM position trajectories.

3.3.3 Classifying trials as successful/unsuccessful

Two clear instances of a subject's chosen control strategy (natural, Momentum-Transfer, or Quasi-Static) failing during sit-to-stand are taking a step and sitting back down. These represent instabilities in a person's motion which could lead to falls. We refer to the trials in which these occurred as *steps* and *sits*, and collectively label them *unsuccessful*. At times, we will also refer to these trials as *failures*. For step trials, the instance of failure is defined as the time when the subject moves the toes of either foot more than 0.0762 m (3 inches) in the anterior-posterior direction from its starting point, as measured by motion capture markers placed on the subject's foot. Although previous studies measured ground reaction forces to identify the onset of steps [158], the distance threshold was used here to utilize motion capture data. For sit failures, the failure instance is defined as the time when both horizontal and vertical velocities of the COM become negative, i.e. the COM begins to move back towards the seat. If a subject both stepped and sat down, failure is defined as the earlier of the two. All foot-shift trials collected according to the parameters specified above were considered successful. Subjects attempting the extreme anterior and posterior foot positions were either unable to initiate the sit-to-stand movement, or violated the parameters of acceptable trials stated earlier. in Sec. 3.3. In either case, these trials were not included in the dataset. If no step or sit is detected, the trial is considered *successful*. In this work, 'successful trials' refers to nominal trials, foot-shift trials, and successful perturbed trials. Note, this scheme classifies trials in which subjects rocked far onto their heels or toes to maintain balance as successful, so long as no step or sit occurred.

3.4 Dynamic modeling

To describe our dynamic sit-to-stand model and trajectory formation, we adopt the following mathematical notation. Let $A \times B$ denote the Cartesian product of sets A and B. Let $\mathbb{R}^n$ be the n-dimensional Euclidean space, and let $\tau \in [0, T] \subset \mathbb{R}$ denote a time drawn from a time interval.

3.4.1 Telescoping Inverted Pendulum Model

We use a Telescoping Inverted Pendulum Model (TIPM, Fig. 3.4) to model each subject's COM motion in the sagittal plane during each sit-to-stand trial [159]. This model is capable of describing the sizable displacements of a subject's COM in the horizontal and vertical directions that occur during sit-to-stand. The TIPM consists of a point mass of mass $m \in \mathbb{R}$ representing the subject's COM. Individualized TIPMs are constructed for each subject by setting m as the subject's mass. Without loss of generality, we assume each sit-to-stand trial starts at time $\tau = 0$.

We define the origin as the initial position of the ball of the subject's foot. Specifically, the origin is defined as the mean initial position of motion capture markers attached to the metatarsophalangeal joints of the subject's left and right feet, as estimated by Visual3D. We let the subscript $(\cdot)_x$ denote a quantity in the anterior-posterior direction, and the subscript $(\cdot)_y$ denote a quantity in the vertical direction. Please note that to make comparisons across trials which may vary in length, we normalize trajectories so that $t \in [0, 1]$. At each time t, we denote the following quantities of the TIPM as:

- Positions: $r_x(t), r_y(t) \in \mathbb{R}$
- Velocities: $v_x(t), v_y(t) \in \mathbb{R}$
- Accelerations: $a_x(t), a_y(t) \in \mathbb{R}$
- Inputs: $u_x(t), u_y(t) \in \mathbb{R}$
- Cable-pull forces: $d_x(t), d_y(t) \in \mathbb{R}$

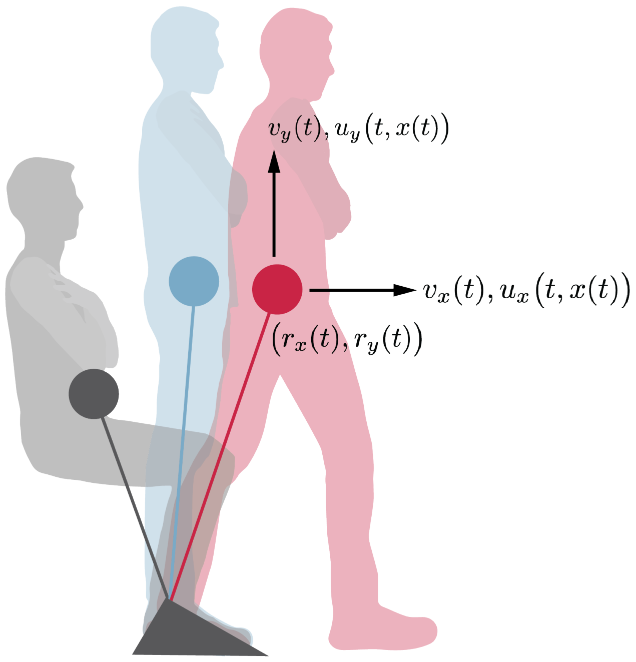


Figure 3.4: Subjects began from a seated position with their arms crossed against their chests (gray). The subject's COM is illustrated by filled circles. Cable-pulls applied to the subject sometimes caused them to step or sit (red); otherwise, the trial was considered to be successful (blue).

Let $\tilde{x} : [0, T] \to X \subset \mathbb{R}^4$ be a state trajectory, so that $\tilde{x}(\tau) \in X$ is the model's state at time $\tau \in [0, T]$:

$$\tilde{x}(\tau) = \begin{bmatrix} \tilde{r}_x(\tau) \\ \tilde{v}_x(\tau) \\ \tilde{r}_y(\tau) \\ \tilde{v}_y(\tau) \end{bmatrix}. \tag{3.1}$$

Let $\tilde{u} : [0, T] \times X \to U \subset \mathbb{R}^2$ be an input trajectory, so that $\tilde{u}(\tau, \tilde{x}(\tau))$ is the input at that time and state:

$$\tilde{u}(\tau, \tilde{x}(\tau)) = \begin{bmatrix} \tilde{u}_x(\tau, \tilde{x}(\tau)) \\ \tilde{u}_y(\tau, \tilde{x}(\tau)) \end{bmatrix}. \tag{3.2}$$

The time derivative of $\tilde{x}(\tau)$ at time τ, denoted $\dot{\tilde{x}}(\tau)$, can be written as:

$$\dot{\tilde{x}}(\tau) = \begin{bmatrix} \tilde{v}_x(\tau) \\ \frac{1}{m}\left(\tilde{u}_x(\tau, \tilde{x}(\tau)) + \tilde{d}_x(\tau)\right) \\ \tilde{v}_y(\tau) \\ \frac{1}{m}\left(\tilde{u}_y(\tau, \tilde{x}(\tau)) + \tilde{d}_y(\tau)\right) - g \end{bmatrix}, \tag{3.3}$$

where g is the gravitational acceleration $9.81 m/s^2$.

3.4.2 Nondimensionalizing time

The time that it takes to complete sit-to-stand varies from trial to trial. Details on how the start and end of each sit-to-stand trial are chosen are provided in Sec. 3.4.3. To make accurate comparisons across trials, we normalize each TIPM state trajectory $\tilde{x}$ by the trial's length, so that each trajectory occurs over the interval $[0, 1]$, which we refer to as 0 to 100 %STS. We introduce a unitless time variable t, related to τ by $t = \frac{\tau}{T}$. This dimensionless variable allows us to introduce a normalized state trajectory $x : [0, 1] \to X$, where

$$x(t) = \begin{bmatrix} \tilde{r}_x(T \cdot t) \\ T \cdot \tilde{v}_x(T \cdot t) \\ \tilde{r}_y(T \cdot t) \\ T \cdot \tilde{v}_y(T \cdot t) \end{bmatrix}. \tag{3.4}$$

We scale accelerations and forces by a factor of T^2 following similar logic.

For the rest of this document, we only consider normalized trajectories, unless explicitly stated. Let r_x, r_y, v_x and v_y refer to normalized position and velocity trajectories, and let a_x, a_y, u_x, u_y, d_x and d_y refer to normalized acceleration and force trajectories.

3.4.3 Determining start and end of sit-to-stand

To segment the continuous kinematic sit-to-stand data into individual trials, an appropriate start and end time for each trial must be chosen. A consistent segmentation method is imperative for making comparisons across trials. Ideally, the start and end of each trial is chosen so that important kinematic features of sit-to-stand are aligned.

We developed the following segmentation procedure, displayed in Fig. 3.5, which is designed to align the peak horizontal accelerations of sit-to-stand trials while being robust to differences in

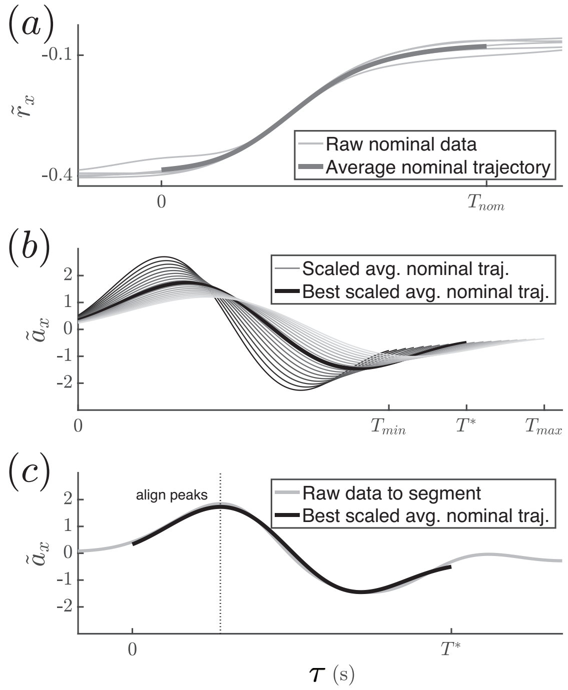


Figure 3.5: This figure illustrates the segmentation procedure. Step 2(b) of the procedure is depicted in Fig. 3.5(a). We take the subject's five nominal trials, shown in light grey, and average them to form a single average nominal trajectory, shown as the thick darker grey line. We choose a start and end time $[0, T_{\text{nom}}]$ for the average nominal trajectory as in Step 2(d). Step 3(a-b) of the procedure is depicted in Fig. 3.5(b). We obtain the horizontal acceleration trajectory of the average nominal trajectory by twice differentiating the position trajectory shown in Fig. 3.5(a). We then iterate over time scales in $[T_{\text{min}}, T_{\text{max}}]$, scaling the trajectory according to Sec. 3.4.2. These scaled trajectories are shown in shades of grey. An example trajectory corresponding to time T^* is shown as the thick black line. Step 3(c-e) of the procedure is depicted in Fig. 3.5(c). The horizontal acceleration trajectory of a trial to be segmented is shown as the thin light grey line. Of the scaled trajectories in Fig. 3.5(b), the one corresponding to time T^*, shown as the thick black line, matches the best. The start and end times of the trial to be segmented are then determined by the endpoints of the overlaid black line.

trial duration. Portions of the procedure utilize a root-mean-square error (RMSE), defined as:

$$RMSE(\hat{y}, y) = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{N}} \tag{3.5}$$

where $\hat{y} = \{\hat{y}_i\}_{i=1}^N, \hat{y}_i \in \mathbb{R}$ and $y = \{y_i\}_{i=1}^N, y_i \in \mathbb{R}$ are two sequences defined by sampling two different trajectories at N times.

The procedure generates an average nominal trial from the subject's five nominal trials, and then uses it as a template to segment the rest of the trials. We segment the natural and Momentum-Transfer strategies' trials using the following procedure, but use a separate procedure for Quasi-Static trials, detailed later on in this section.

1. Manually segment each trial, starting several seconds before the subject begins to stand and ending several seconds after they have reached standing.

2. To create an average nominal trial:

    a. Take the manual segmentations of the subject's five nominal trials, and align their COM position trajectories such that the times at which the peak horizontal COM velocities occur are coincident.

    b. Average the subject's five nominal trials together to form a single average nominal COM position trajectory.

    c. Numerically differentiate the average nominal COM position trajectories to obtain velocity and acceleration trajectories.

    d. Choose the start time ($\tau = 0$) of the average nominal trajectory as when the horizontal COM acceleration first exceeds 20% of its max. Choose the end time ($\tau = T_{\text{nom}}$) as when the vertical COM position first exceeds 99% of its max.

    e. The average nominal trajectory is now defined over $[0, T_{\text{nom}}]$. Follow the scaling laws in Sec. 3.4.2 so that it is defined over $[0, 1]$.

3. To segment each trial:

    a. Iterate over times $T \in [T_{\text{min}}, T_{\text{max}}]$, where the interval is discretized into 100 samples. Let $T_{\text{min}} = 0.75s$ and $T_{\text{max}} = 1.5T_{\text{nom}}$.

    b. Rescale the average nominal trajectory to be defined over $[0, T]$.

    c. Automatically align the scaled average nominal trial and the current trial so that the time at which the peak horizontal COM accelerations occur are coincident.

    d. The trajectories overlap in a window of length T seconds. Compute the RMSE between the horizontal COM acceleration trajectories over this window.

    e. Find T that minimizes this RMSE (to accurately segment cable pull trials despite the effects of perturbation, we only minimize the pre-perturbation RMSE for cable pull trials).

Because the Quasi-Static strategy utilizes minimal forward momentum, it lacks a distinct peak in its horizontal COM acceleration trajectories when compared to the natural and Momentum-Transfer strategies. This is shown in Fig. 3.10. Therefore, we developed a separate segmentation procedure for the Quasi-Static strategy, where Steps 1 and 2 remain the same as before:

3. To segment each Quasi-Static trial:

    a. Iterate over times $T \in [T_{\text{min}}, T_{\text{max}}]$, where the interval is discretized into 100 samples. Let $T_{\text{min}} = 0.75s$ and $T_{\text{max}} = 1.5T_{\text{nom}}$.

    b. Rescale the average nominal trajectory to be defined over $[0, T]$.

    c. Manually align the scaled average nominal trial and the current trial so that the time at which the peak vertical COM velocities occur are coincident.

    d. The trajectories overlap in a window of length T seconds. Manually choose T that minimizes the RMSE between the scaled average nominal vertical COM velocity trajectory and the current trial's COM velocity trajectory. (Prioritize the pre-perturbation difference for cable pull trials, to accurately segment cable pull trials despite the effects of perturbation.)

3.5 Controller Models

The controller model estimates the control input $u(t, x(t))$ at a given time t and state $x(t)$. In many previous studies, researchers have modelled human controllers as an open-loop "feedforward" component plus a closed-loop linear "feedback" component [160, 161, 92], thus finding a single controller that best models the data. Because the form of the controller employed by humans is unknown, it is difficult to assert with confidence that a single controller adequately captures human behavior. We propose that a more robust controller model may return a set of possible inputs rather than a single input at a given time and a state. One way to achieve this is to bound the possible inputs at each time and state, and assume that the human could use any input within these bounds. We refer to this controller model as the *Input Bounds* controller.

In this subsection, we describe and compare two existing controller models as well as our proposed controller model. All three controller models are used to compute Stability Basins that are compared during the validation phase described in Sec. 3.6.7. Before we describe these controller models in mathematical detail, we briefly describe their qualitative properties. The controllers are compared and contrasted in Fig. 3.6.

The controller models estimate the control input $u(t, x(t))$ at a given time t and state $x(t)$. In many previous studies, researchers have modelled human controllers as an open-loop "feedforward" component plus a closed-loop linear "feedback" component [160, 161, 92], thus finding a single controller that best models the data. Following this paradigm, we develop an LQR controller model and a traditional feedforward plus feedback controller model (FF+FB). The main differences between these controllers are the specifics of their construction, detailed in Sec. 3.5.2 and Sec. 3.5.3. The LQR controller model is constructed by solving an optimal control problem to provide feedback about an average nominal trajectory, while the FF+FB controller model is constructed via linear regression. Both of these controller models return a single input at a given time and state.

Because the form of the controller employed by humans is unknown, it is difficult to assert with confidence that a single controller adequately captures human behavior. We propose that a more robust controller model may return a set of possible inputs rather than a single input at a given time and a state. One way to achieve this is to bound the possible inputs at each time and state, and assume that the human could use any input within these bounds. We refer to this controller model as the *Input Bounds* controller.

The Input Bounds represent the range of time-varying inputs that are expected under a given sit-to-stand strategy. For example, consider a subject's natural sit-to-stand strategy. At the beginning of the motion, we expect the subject to apply a large positive horizontal input to propel themselves forwards, while applying little positive vertical input. Halfway through the motion, the subject may apply a negative horizontal input to slow their forward velocity, as well as a large positive vertical input to rise towards standing. Therefore, the expected inputs applied by the subject are time-dependent. Moreover, because there is some variability in the way the subject performs sit-to-stand from trial to trial, we expect the inputs to fall within some range of values. The Input Bounds controller captures this variability by developing time-dependent upper and lower bounds on the allowable inputs.

Biomechanical constraints, such as ground reaction force limits or joint torque limits, constrain the maximum inputs that a subject can apply during sit-to-stand under any strategy. We want to be clear that the Input Bounds *do not* represent the range of all inputs that are biomechanically feasible, but rather the subset of inputs that represent the range expected under a given strategy. For example, consider a scenario in which a person begins standing up using their natural sit-to-stand strategy, but then stops halfway and remains in a crouched position. Certainly, the person can achieve this performance without violating biomechanical constraints. However, the person will have deviated from their natural sit-to-stand strategy, in which they normally stand all the way up. Accordingly, the input that they applied would have exited the Input Bounds we have developed for their natural strategy at the time that they decided to remain in a crouched position. For this reason, we develop the Input Bounds from observations of sit-to-stand rather than from biomechanical constraints.

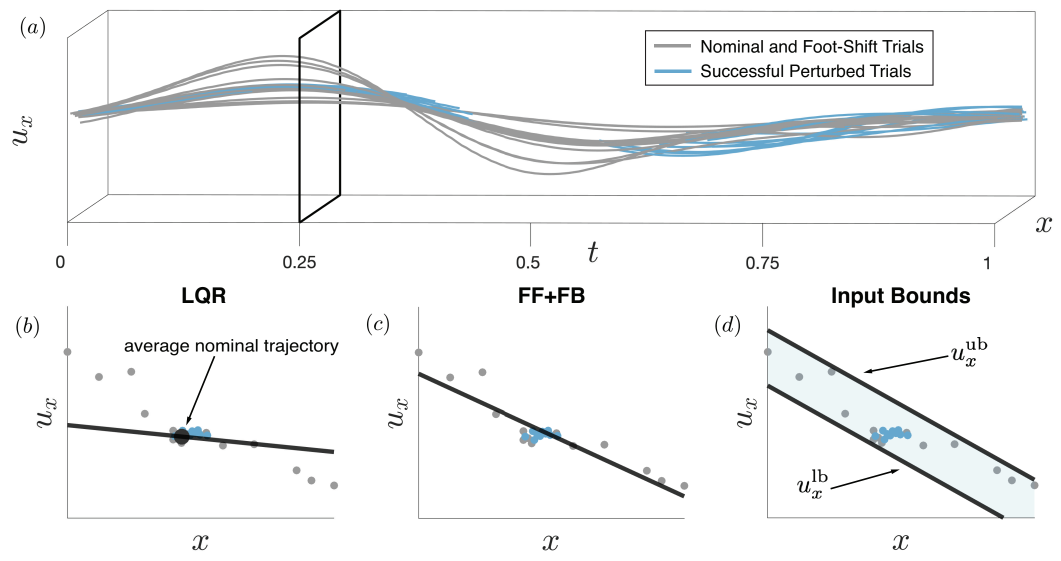


Figure 3.6: These figures show how we construct the controller models from data. The process is shown for u_x, and is identical for u_y. In Fig. 3.6(a), u_x evolves as a function of time t and state x, which has been depicted as a single dimension for this illustration. The grey lines represent the input trajectories of nominal and foot-shift trials, while the blue lines represent the input trajectories of perturbed successful trials. Note that we have not plotted the portions of the successful perturbed trajectories when the perturbations were active, because these segments are not used to train the controller models. An arbitrary time $t = 0.25$ is illustrated by the black outline in Fig. 3.6(a). The observed inputs from nominal, foot-shift, and perturbed successful trials at that time are plotted as points in Fig. 3.6(b - d). Figure 3.6(b) shows that the LQR controller utilizes linear feedback about a nominal trajectory. Because the LQR controller is generated by minimizing a cost function specified by the matrices Q and R, its feedback matrix does not necessarily coincide with a line of best fit. Figure 3.6(c) illustrates that the FF+FB controller is given by a line of best fit. Figure 3.6(d) shows how the upper and lower bounds $u_x^{\mathrm{lb}}(t, x(t))$ and $u_x^{\mathrm{ub}}(t, x(t))$ are parallel linear functions of state that encompass all of the successful inputs.

3.5.1 Generating training data

Modelling human motion requires the LQR, FF+FB, and Input Bounds controller models to be estimated from observed data. Modelling human motion requires the controller models to be estimated from observed data. In this work, we use kinematic observations of successful trials (i.e. nominal, foot-shift, and successful perturbed trials) to generate a dataset on which to train each control model. Note that the dataset on which each controller model is trained is identical. We fit the controllers separately for each subject's natural, Momentum-Transfer, and Quasi-Static control strategies.

Given a strategy, we first compute the inputs for each successful trial within that strategy via inverse dynamics using (3.3) [162, Chapter 5]. Let $x_i(t)$ and $u_i(t)$ denote the state and input at time t for the i^{th} successful trial computed via inverse dynamics, and let S denote the set of successful trials within a given strategy. Given a time t, we will frequently use the notation

$$\sum_{i \in S} \left(u(t, x_i(t)) - u_i(t)\right)^2 \tag{3.6}$$

which means "the squared error of the estimated control input $u(t, x_i(t))$ from each observed control input $u_i(t)$, summed over all successful trials within a given strategy." Because many successful trials within a sit-to-stand strategy involved cable-pull perturbations, we make the following remark:

Remark 3.5.1. *When computing $u_i(t)$ via inverse dynamics for successful perturbed trials, we discard the portion of the trajectory during which the perturbation was active. This is possible because the perturbations were time-synchronized with the motion capture data.*

3.5.2 LQR controller

Shia et al. proposed to model sit-to-stand using an LQR controller about a nominal sit-to-stand trajectory [92]. To build this controller, we first form a single average nominal trajectory $\bar{x}$ for each of a subject's sit-to-stand control strategies by taking the mean of the five nominal sit-to-stand trajectories. We generate an open loop controller, $u_{ol} : [0, 1] \to U \subset \mathbb{R}^2$, for each average nominal trajectory via inverse dynamics. Then, we use LQR to design a linear feedback controller about the average nominal trajectory. We specify the quadratic cost function for LQR by a state weighting matrix $Q \in \mathbb{R}^{4\times4}$ and input weighting matrix $R \in \mathbb{R}^{2\times2}$, which are used to generate the feedback matrix $K(t) \in \mathbb{R}^{2\times4}$. [163, Chapter 16] At a given time and state, the input from the LQR controller can then be written as:

$$u(t, x(t)) = u_{ol}(t) - K(t)(x(t) - \bar{x}(t)) \tag{3.7}$$

We choose the Q and R that produces a controller that most closely fits the observed data. To do so, we wrap the choice of Q and R inside an optimization procedure that seeks to minimize the error between the LQR controller model and the observed control inputs over the time horizon $[0, 1]$:

$$\min_{Q,R} \sum_{i \in S} \int_0^1 \left(u(t, x_i(t)) - u_i(t)\right)^2 dt \tag{3.8}$$

As in (3.6), S represents the set of observed successful trials within a sit-to-stand strategy, and

$x_i(t)$ and $u_i(t)$ correspond to the observed states and input of the i^{th} trial at time t. The integral is approximated using 200 time steps. We specify the space over which Q is optimized:

$$Q \in \{M \in \mathbb{R}^{4\times 4} | M_{i,j} = 0\,\forall i \neq j,\ M_{1,1} = 1,\ M_{2,2} \in [0.1, 100],\ M_{3,3} \in [0.1, 100],\ M_{4,4} \in [0.1, 100]\} \tag{3.9}$$

and the space over which R is optimized:

$$R \in \{M \in \mathbb{R}^{2\times 2} | M_{i,j} = 0\,\forall i \neq j,\ M_{1,1} \in [1e{-5}, 1e{-2}],\ M_{2,2} \in [1e{-6}, 1e{-1}]\} \tag{3.10}$$

Q and R are constant diagonal matrices, where the spaces over each which is optimized were chosen to restrict the relative costs of states versus inputs, the relative costs between states, and the relative costs between inputs. In particular, no state can cost more than 100 times any other state, no input can cost more than 10 times another input, and the relative scale between states and inputs was chosen based on the expected magnitudes of states and inputs.

3.5.3 FF + FB controller

We also test the efficacy of a traditional feedforward plus feedback controller. Let the input $u(t, x(t))$ at some time t and state $x(t)$ be described by the state feedback matrix $K(t) \in \mathbb{R}^{2\times 4}$ and a feedforward component $\text{ff}(t)$:

$$u(t, x(t)) = \text{ff}(t) - K(t)x(t) \tag{3.11}$$

To generate $K(t)$ and $\text{ff}(t)$ from data, we use linear least squares to solve the following program:

$$\min_{\text{ff}(t), K(t)} \sum_{i \in S} \left(u(t, x_i(t)) - u_i(t)\right)^2 \tag{3.12}$$

As in (3.6), S represents the set of observed successful trials within a sit-to-stand strategy, and $x_i(t)$ and $u_i(t)$ correspond to the observed states and input of the i^{th} trial at time t.

3.5.4 Input Bounds controller

The proposed Input Bounds controller differs from both the LQR and FF+FB controllers by returning a set of inputs at a given time and state, rather than a single point. To define the Input Bounds controller, we develop bounds on the TIPM control input. We denote a lower bound on the control input as u^{lb}, and an upper bound on the control input as u^{ub}. Note that

$$u^{\text{lb}} \leq u \leq u^{\text{ub}} \tag{3.13}$$

where the inequality is taken element-wise. We present this approach more formally in the following definition:

Definition 3.5.2. *For the Input Bounds controller, we model the inputs u to the TIPM as a bounded component plus a linear feedback component, where the bounded component is drawn from a bounded set. Specifically, given time t and state $x(t)$, we define the bounds in* (3.13) *as*

$$u^{\text{lb}}(t, x(t)) = b^{\text{lb}}(t) - K(t)x(t) \tag{3.14}$$

$$u^{\text{ub}}(t, x(t)) = b^{\text{ub}}(t) - K(t)x(t) \tag{3.15}$$

where $b^{\text{lb}}(t)$ and $b^{\text{ub}}(t) \in \mathbb{R}^2$ are lower and upper bounds on the bounded component, and $K(t) \in \mathbb{R}^{2\times 4}$ is a matrix of linear feedback gains.

We generate the parameters of the Input Bounds controller b^{lb}, b^{ub}, and K from data by solving a constrained linear least squares problem:

$$\begin{aligned} \min_{b^{\text{lb}}(t), b^{\text{ub}}(t), K(t)} \quad & \sum_{i\in S} \left(u^{\text{lb}}(t, x_i(t)) - u_i(t)\right)^2 + \left(u^{\text{ub}}(t, x_i(t)) - u_i(t)\right)^2 \\ \text{s.t.} \quad & u^{\text{lb}}(t, x_i(t)) \leq u_i(t) \qquad \forall i \in S, \\ & u^{\text{ub}}(t, x_i(t)) \geq u_i(t) \qquad \forall i \in S \end{aligned} \tag{3.16}$$

where the inequalities are understood element-wise. This is a quadratic program, which can be solved to global optimality rapidly and efficiently.

In summary, at each instance in time, we compute the range of control inputs of a certain form that can be applied while explaining the data we observed, thereby modelling a range of inputs a human may use to recover from perturbation. Our method of Stability Basin computation, detailed in Sec. 3.6.2, is flexible to this type of controller specification, and in particular, we verify that this approach is not too conservative in Sec. 3.7. In effect, the Input Bounds controller model allows us to compute Stability Basins with a set of hypothetical controllers, instead of just one.

3.6 Computing and Evaluating Stability Basins

Three elements are required to form the Stability Basin for each of an individual's sit-to-stand control strategies:

1. The TIPM dynamics (3.3).

2. A controller model for the TIPM (e.g. (3.13)).

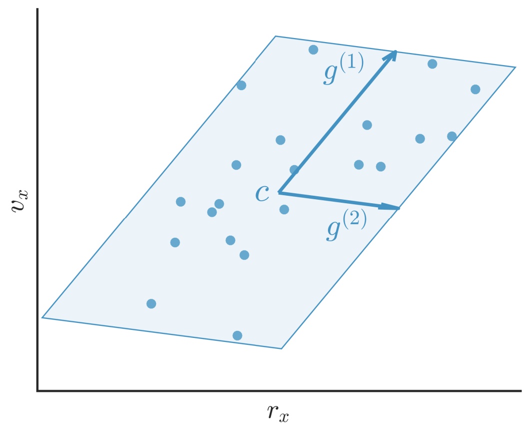


Figure 3.7: This figure illustrates how X_T (the target set) is formed from data for a given subject and strategy. The states (r_x and v_x) at 100% STS of each observed successful trial within a strategy are illustrated as blue dots. An example X_T represented by the shaded blue zonotope contains all of the states observed to lead to standing. The zonotope is parameterized by its center c and generator vectors $g^{(1)}$ and $g^{(2)}$. This example depicts a 2-dimensional X_T, but in reality they are 4-dimensional and encompass both horizontal and vertical components of the state space.

3. A *target set*, X_T, that encapsulates all model states observed to lead to successful standing.

Given these elements, we define the Stability Basin as follows:

Definition 3.6.1. *The Stability Basin* $\subset [0,1] \times X$ *is a subset of times (i.e., 0 - 100%STS) and model states from which a trajectory of the model obeying the dynamics* (3.3) *and controller model will arrive at the target set* X_T*.*

In the rest of this subsection, we explain how we compute the target set X_T from data. Then, we use this object in conjunction with a controller from Sec. 3.5 to generate the Stability Basins.

3.6.1 Generating the target set

The goal of the sit-to-stand motion is to arrive at a standing configuration. We define the target set X_T as a set that encompasses the states at time $t = 1$ observed to lead to successful standing. Note, the target set X_T does not represent quiet standing itself, which we consider a distinct task. We define $X_T \subset X$ as a zonotope (Sec. 1.5.2) that encompasses the final states (i.e., the states at $t = 1$) of all successful trials observed for a given sit-to-stand strategy. Specifically, X_T is a zonotope with 4 generators computed from the final states as proposed by Stursberg [164, Section 3], where each generator is expanded by 5% to avoid observed states lying on the edge of the set. The target set generation process is depicted in Fig. 3.7.

3.6.2 Computing Stability Basins via reachability analysis

The reader may notice that the Stability Basin (Defn. 3.6.1) is defined as a set of states through time that satisfy some dynamics specified by an ordinary differential equation (ODE). As we are interested in finding all possible solutions of the ODE, subject to constraints, we use a method called reachability analysis to compute the Stability Basins. Reachability analysis is introduced in Sec. 1.5.2.

To compute the Stability Basins, we use an open-source zonotope-based reachability toolbox called CORA [102, 104]. For an explanation of CORA and zonotopes, also see Sec. 1.5.2. CORA represents the Stability Basin as a zonotope at each of a finite collection of *time steps*, which are subintervals of the interval $[0, 1]$ of length Δt. Each Stability Basin was formed using a time step Δt of 0.005 (i.e., 0.5%STS), so that 200 zonotopes represent the Stability Basin over the interval $[0, 1]$. Although CORA chooses the exact number of generators to use for each zonotope, we set the maximum number of generators as 800. Generally, CORA expects the inputs to the system to be functions of time, allowing the feedforward inputs and gain matrices for the LQR and FF+FB controllers to be specified at each time step. However, CORA also allows the input to the system at each time step to be drawn from a set, allowing us to use the Input Bounds controller model and let the control input be defined as in (3.14) and (3.15). CORA assumes the input at a given time and state can take any value within the Input Bounds to create the Stability Basin. Because the Stability Basin is the set of states through time that arrive at the final target set X_T, it can be computed as a backwards reachable set. In practice, this means we treat X_T as an initial set, and use CORA to flow the set *backwards* in time under the negative of the dynamics (3.3) and the controller model. Stability Basins were generated on a laptop computer with a 2.7 GHz Intel Core i7 processor. Stability Basins computed using the Input Bounds controller take 0.76 ± 0.014 seconds to compute.

The computed Stability Basins provide the following theoretical guarantee, which we reference in Sec. 1.4 and in the dissertation summary in Fig. 1.2:

Theorem 3.6.2. *If a state $x(t)$ at time t lies outside the computed Stability Basin, and the system obeys the TIPM dynamics* (3.3) *and controller model (e.g. Input Bounds* (3.16)*), then no valid trajectory exists which takes $x(t)$ to the target set X_T. More specifically, $\nexists z : [t, T] \to X$ s.t. $z(T) \in X_T, z(t) = x(t)$ and z satisfies the dynamics and controller model.*

Proof. CORA computes overapproximations of reachable sets, as introduced in Sec. 1.5.2. By definition (Def. 3.6.1), any state trajectory which obeys the system dynamics and controller model and ends in the target set is contained in the Stability Basin. The computed Stability Basins produced by CORA overapproximate the theoretical Stability Basins. If a state $x(t)$ at time t is outside the computed Stability Basin, it is necessarily outside of the theoretical Stability Basin. Therefore, it is impossible to form a valid trajectory which takes $x(t)$ to the target set. ■

This guarantee provides a condition for predicting failure during sit-to-stand. More colloquially, if a state $x(t)$ at time t is outside the computed Stability Basins then there is no input the person can apply to steer themselves to the target set. This means that they will either have to change their control strategy to avoid a fall, e.g. by stepping or sitting, or perform an action that is not included in our model of sit-to-stand.

3.6.3 Naive method for computing Stability Basins

Stability can also be estimated from observed perturbed trials by simply drawing a volume around the state trajectories of the observed successful trials. This method does not utilize reachability or a controller model, and relies on state trajectories alone. Here, we apply this method by generating a 4-dimensional zonotope at each time step that encloses all of the observed successful states. This is the same procedure that we employ to generate the target set X_T (described in Sec. 3.6.2 and illustrated in Fig. 3.7) applied at each instance in time over the course of the motion.

3.6.4 Evaluating Stability Basin accuracy

We hypothesize that the Stability Basins can predict when a subject must depart from their control strategy in response to perturbation to avoid falling. Previously in Sec. 3.3.3, we defined the failure of a control strategy as stepping or sitting back down and explained how we determine the onset of stepping or sitting. We now use the Stability Basins to predict whether or not a strategy failure will occur by checking if a sit-to-stand trajectory exits the Stability Basin at any point. Finally, we detail the evaluation procedure we used to test the accuracy of the Stability Basins' predictions.

3.6.5 Defining onset of failure

To test the predictive power of the Stability Basins, we first identify the onset of failure (as previously described in Sec. 3.3) during a sit-to-stand movement. We denote $t_f \in [0, 1]$ as the time of failure onset, and will test whether the Stability Basins predict failure prior to that time. Given an observed trajectory x_i from a sit-to-stand trial, we use a linear program to determine whether that trajectory lies inside the Stability Basin at each time step. Because we are using a procedure based on an average nominal trial for aligning and segmenting trials (detailed in Sec. 3.4.3), it is possible for the onset of stepping or sitting to occur after the trial end time $t = 1$. The end times chosen by the segmentation procedure do not necessarily imply that the subject has reached standing, but are just a means to achieve a consistent segmentation of the data. We consider a trial with a step or sit occurring later than the trial's end to be unsuccessful, and set $t_f = 1$ in this case.

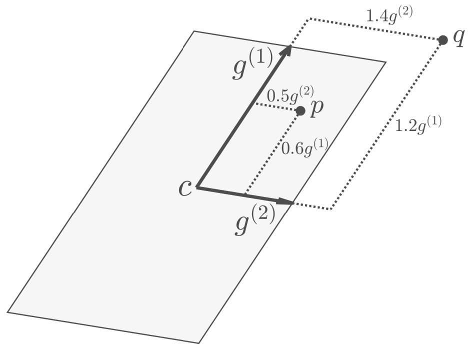


Figure 3.8: This figure illustrates how to check if a point is inside or outside of a zonotope. An example zonotope (shaded grey) is parameterized by its center c and generator vectors $g^{(1)}$ and $g^{(2)}$. A test point p is contained within the zonotope because the maximum absolute value of the coefficients on $g^{(1)}$ and $g^{(2)}$ is less than or equal to 1. Another test point q lies outside of the zonotope because the maximum absolute value of the coefficients on $g^{(1)}$ and $g^{(2)}$ is greater than 1.

3.6.6 Checking trajectories

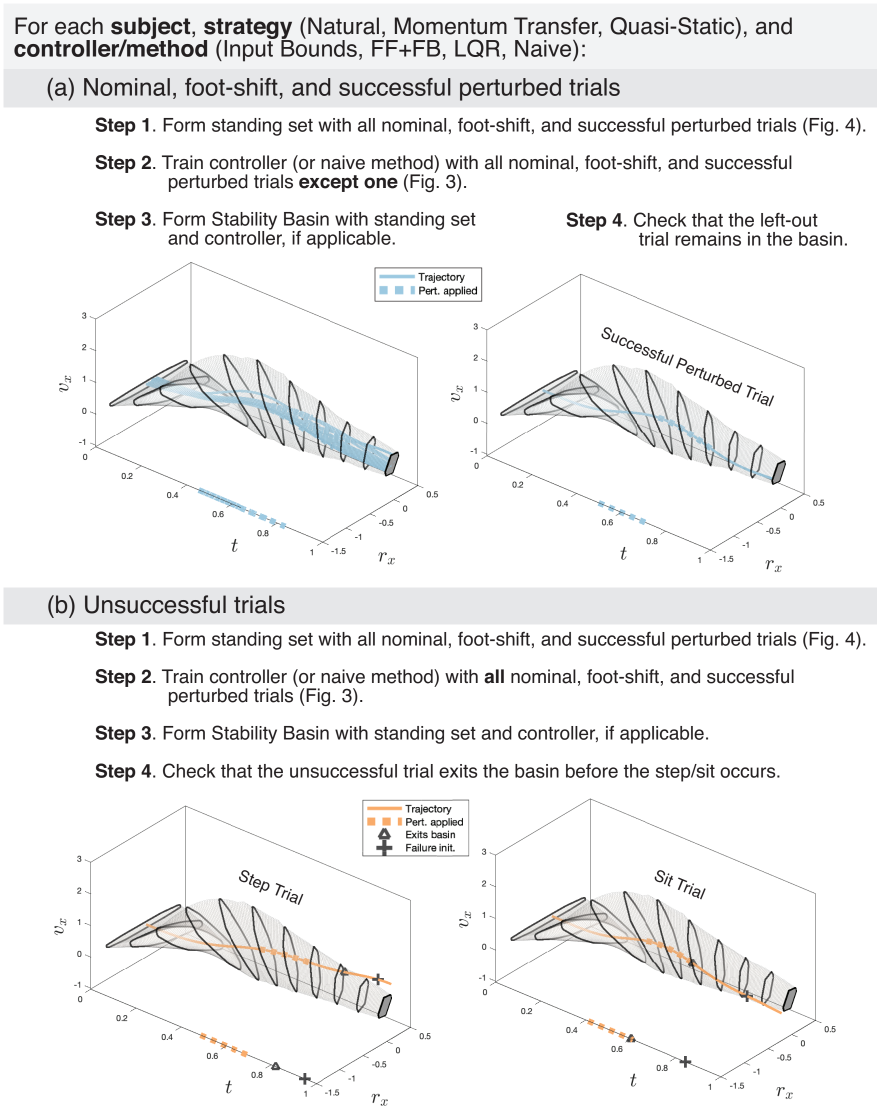
Given an observed trajectory x_i from a sit-to-stand trial, we use a linear program to determine whether that trajectory lies inside the Stability Basin at each time step. Recall that each zonotope $Z^{(t)}$ representing the Stability Basin is parameterized by its center $c^{(t)} \in \mathbb{R}^4$ and generator matrix $G^{(t)} \in \mathbb{R}^{4\times p}$. At each time t, we represent the state $x_i(t)$ as a linear combination of the generator matrix's columns, and use a linear program to find the representation that has the smallest maximum coefficient $\beta^{(t)}_{\text{max}} \in \mathbb{R}$:

$$\begin{aligned} \min_{\beta^{(t)}_{\text{max}} \in \mathbb{R}, \beta^{(t)} \in \mathbb{R}^p} \quad & \beta^{(t)}_{\text{max}} \\ \text{s.t.} \quad & G^{(t)}\beta^{(t)} = x_i(t) - c^{(t)}, \\ & |\beta^{(t)}| \leq \beta^{(t)}_{\text{max}} \end{aligned} \tag{3.17}$$

where the absolute value and inequalities are applied element-wise. If $\beta^{(t)}_{\text{max}}$ is less than or equal to 1, the point $x_i(t)$ is inside the Stability Basin at that time step, depicted in Fig. 3.8.

3.6.7 Evaluation procedure

The Stability Basins are used to predict the success or failure of every trial within each sit-to-stand strategy (natural, Momentum-Transfer, and Quasi-Static). The accuracy of each Stability Basin is then tested by comparing its predictions to the experimentally observed outcomes of each sit-to-stand trial. Given the state trajectory x_i observed for a sit-to-stand trial, the linear programs


For each **subject**, **strategy** (Natural, Momentum Transfer, Quasi-Static), and **controller/method** (Input Bounds, FF+FB, LQR, Naive):

(a) Nominal, foot-shift, and successful perturbed trials

Step 1. Form standing set with all nominal, foot-shift, and successful perturbed trials (Fig. 4).

Step 2. Train controller (or naive method) with all nominal, foot-shift, and successful perturbed trials **except one** (Fig. 3).

Step 3. Form Stability Basin with standing set and controller, if applicable.

Step 4. Check that the left-out trial remains in the basin.

(b) Unsuccessful trials

Step 1. Form standing set with all nominal, foot-shift, and successful perturbed trials (Fig. 4).

Step 2. Train controller (or naive method) with **all** nominal, foot-shift, and successful perturbed trials (Fig. 3).

Step 3. Form Stability Basin with standing set and controller, if applicable.

Step 4. Check that the unsuccessful trial exits the basin before the step/sit occurs.

Figure 3.9: An outline of the Stability Basin validation with examples provided by the computed Stability Basin for subject ID 1's natural strategy under the Input Bounds controller. The horizontal projection of the Stability Basin is represented as the region encapsulated by the light grey borders. The projection of the target set X_T is shown as the dark grey region on the right side of each plot. The state trajectories of all of subject ID 1's successful natural strategy trials *except one* are used to construct the Stability Basin on which the left-out trial is tested, as shown in Fig. 3.9(a). Then all of subject ID 1's successful natural strategy trials are used to construct the Stability Basin on which the unsuccessful trials are tested, as shown in Fig. 3.9(b). State trajectories of a step and a sit exit the basin before the onset of failure. As detailed in Sec. 3.3.3, we define step initiation as the time when the toes of either foot move more than 0.0762 m (3 inches) in the anterior-posterior direction, and sit initiation as the time at which both v_x and v_y become negative.

described in (3.17) are used to predict success/failure. If a state trajectory x_i remains inside the

Stability Basin at each time step, the trial is predicted to be successful. If x_i exits the Stability Basin at any time step, the trial is predicted to fail. To ensure that the Stability Basins' predictions are made fairly, we make the following restrictions:

1. For cable pull trials, the prediction uses only the portion of the Stability Basin after the onset of perturbation.

2. For trials in which failure was observed, the prediction uses only the portion of the Stability Basin before failure occurred (i.e., $t \leq t_f$).

We evaulate the Stability Basins formed using the LQR, FF+FB, and Input Bounds controllers, as well as the naive method in Sec. 3.6.3. We perform separate types of evaluation for successful trials versus unsuccessful trials. For successful trials, we employ a *leave-one-out* procedure to avoid training and testing on the same data. After forming the target set, we leave one successful trial out of the controller (LQR, FF+FB, Input Bounds) training data in Sec. 3.5.1. For the naive method, one successful trial is left out of the zonotope generation at each time step. We then construct a Stability Basin using each controller/method to test the trial that was left out. We then construct a Stability Basin to test the trial that was left out. We repeat this procedure until all successful trials have been tested.

For unsuccessful trials, we compute a single Stability Basin formed using all observed successful trials. We use this single Stability Basin to predict the outcome of each unsuccessful trial within a given strategy. An outline of the validation procedure is provided in Fig. 3.9.

To be clear, we do not follow a leave-one-out procedure when generating the target set X_T from data. As displayed in Fig. 3.7, the target set generation is similar to taking the convex hull of a set of 4-dimensional points. If we follow a leave-one-out procedure when generating the target set, then the points on the boundary of the original set will likely be on the outside of the new set when they are left out. This would mean that many successful trials whose final states lie on the edge of the original target set necessarily exit the Stability Basins, and therefore are automatically misclassified when following this leave-one-out procedure. For these reasons, this procedure would serve to validate the robustness of the target set generation, rather than the Stability Basins themselves. However, collecting a larger number of samples should decrease the effect of missing data on the target set generation.

3.7 Results

The subjects performed sit-to-stand trajectories with similar maximum and minimum accelerations for each control strategy as the (different) subjects examined by Shia [92]. Even though subjects

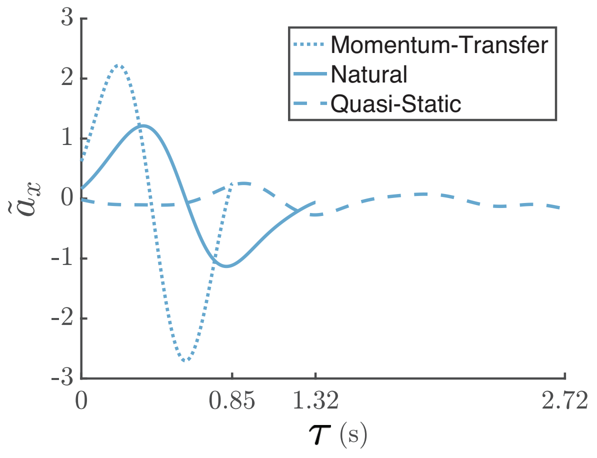


Figure 3.10: This figure shows example nominal unnormalized horizontal acceleration trajectories for each sit-to-stand strategy for subject ID 8.

were allowed to determine the speed with which each sit-to-stand was performed, each subject's natural, Momentum-Transfer, and Quasi-Static sit-to-stand strategies showed distinct kinematic characteristics, depicted in Fig. 3.10. In particular, the maximum and minimum horizontal accelerations of the COM for the Momentum-Transfer ([-2.1, 2.2] m/s^2) and Quasi-Static ([-0.8, 0.6] m/s^2) strategies were significantly different from each other as well as from the natural strategy ([-1.3, 1.5] m/s^2), with p-values < 0.001 when comparing the successful trials within any two strategies. While the duration of natural and Momentum-Transfer trials were similar (average times of 1.25s and 1.13s), Quasi-Static trials lasted longer (average 2.23s), due in part to the low observed COM accelerations. The Stability Basin shape and volume are highly dependent on the kinematic and temporal characteristics corresponding to the strategy selected, as in previous work [92]. When using the Momentum-Transfer strategy during the foot-shift trials, subjects were able to stand with their feet further forward (max. 0.199 ± 0.0699 m) than when using Quasi-Static (max. 0.0417 ± 0.0307 m).

Cable pull perturbations during sit-to-stand frequently induced a step or sit to avoid falling. Out of 591 cable pull trials, we observed failure in 198, or 33.5%. Specifically, we observed 42 steps and 29 sits for the natural strategy, 40 steps and 19 sits for Momentum-Transfer, and 39 steps and 29 sits for Quasi-Static. Fewer sits were observed for the Momentum-Transfer strategy than in both the natural and Quasi-Static strategies, likely due to the larger forward momentum at seatoff for the Momentum-Transfer strategy.

To assess the accuracy of each Stability Basin, we compared the predictions to the experimental observation for each trial according to the procedure described in Sec. 3.6.4. An example Stability Basin is shown in Fig. 3.12 with a step trial. The aggregate prediction rates of the Stability Basins

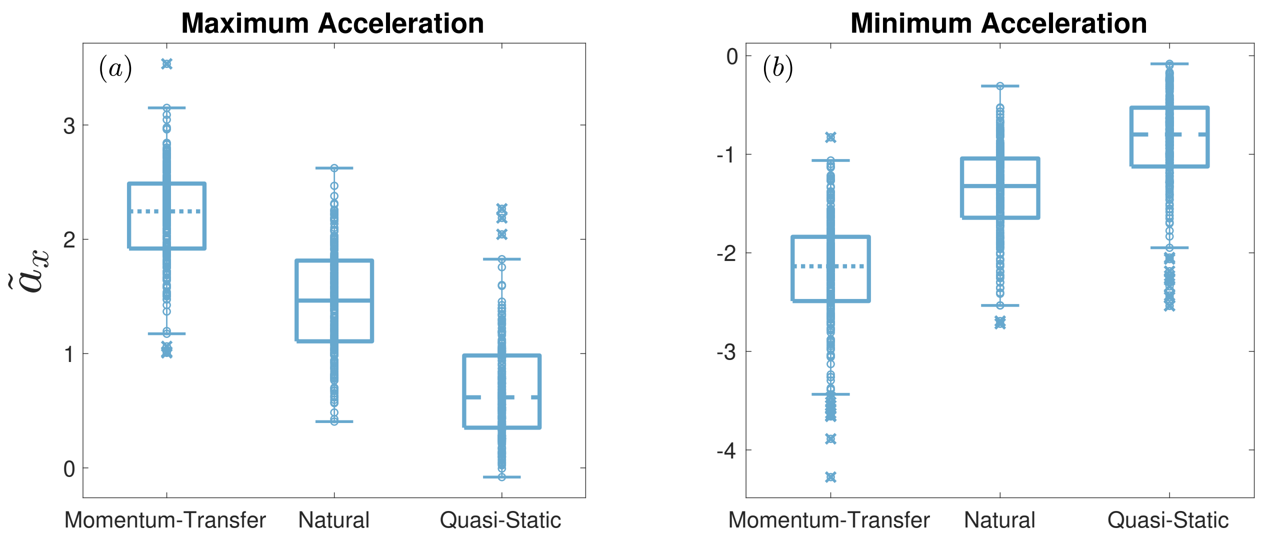


Figure 3.11: Maximum (Fig. 3.11(a)) and minimum (Fig. 3.11(b)) horizontal accelerations are significantly different between sit-to-stand strategies, with $p < 0.001$ when comparing the maximums or minimums of any two strategies under a two-sample t-test. Only successful trials are used in this analysis. For each strategy, each small circle represents data from a single sit-to-stand trial. The central mark within the box plot indicates the median horizontal acceleration. The bottom and top edges of the box indicate the the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme horizontal accelerations not considered outliers, and the outliers are plotted individually as small circles with an 'x' through them.

Table 3.1: Statistics collected from the perturbative sit-to-stand experiment are aggregated across subjects and summarized below. Note that the onset of failure (Step %STS and Sit %STS) often occurred after the defined trial end ($t = 1$), which is discussed in Sec. 3.6.5. Supplementary material that reports these statistics for individual subjects is available online.

Sit-to-Stand Strategy	Total	Cable Pull	Steps	Sits	Trial Time (s)	Pert. Onset (%STS)	Step (%STS)	Sit (%STS)
Natural	326	197	42	29	1.25 ± 0.23	0.50 ± 0.10	1.07 ± 0.21	0.96 ± 0.15
Momentum-Transfer	322	198	40	19	1.13 ± 0.21	0.47 ± 0.11	1.14 ± 0.33	0.99 ± 0.15
Quasi-Static	300	196	39	29	2.23 ± 0.89	0.51 ± 0.17	0.95 ± 0.21	0.85 ± 0.20

formed using the Input Bounds controller for each sit-to-stand strategy across subjects are given in Tab. 3.2. Across sit-to-stand strategies, nominal, foot-shift, and successful perturbed trials remain inside the basin 95.73% of the time. Of the failure trials, 90.91% of steps are predicted prior to onset, while 92.21% of sits are predicted prior to onset. There was no consistent effect of perturbation force on the predictive power of the method, detailed in Tab. 3.3. The Input Bounds produced the most accurate Stability Basins compared to other controller models [93].

The Input Bounds produced the most accurate Stability Basins compared to other controller models [93]. Overall, the Input Bounds Stability Basins' failure predictions are over 90% accurate; only 17 out of 198 unsuccessful trials are incorrectly predicted to be successful. To understand

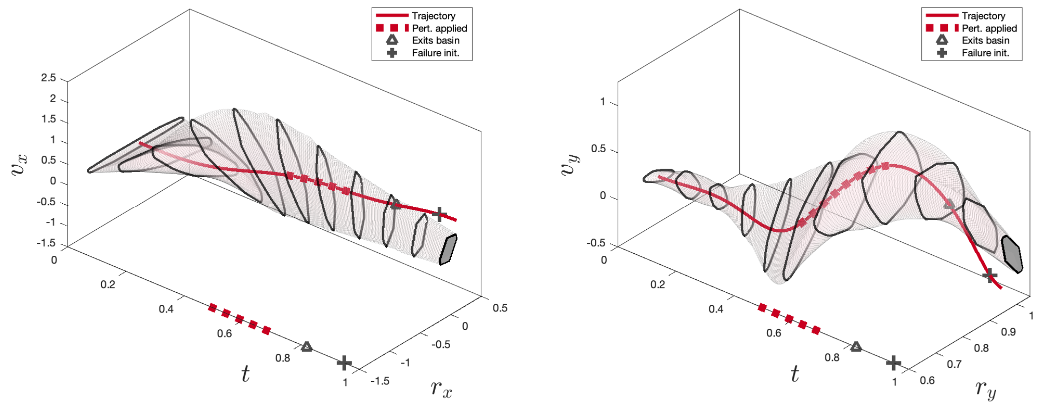


Figure 3.12: An example Stability Basin, with projections of horizontal states on the left and vertical states on the right. The Stability Basin (light gray) is computed as the backwards reachable set of the target set (dark gray, right sides of plots), where black rings are used to accentuate the shape of the basins. A trajectory from a trial in which the subject was pulled forward is shown in red. The thick dashed line indicates the time of perturbation application. A triangle marker indicates the time which the trajectory first exits the Stability Basin, which occurs before failure is initiated, denoted by the cross marker.

Table 3.2: The accuracy of the Stability Basins generated using the Input Bounds controller model are reported below. This table reports a tally of the correct predictions for each sit-to-stand strategy and trial type across subjects.

Sit-to-Stand Strategy	Successful Trials	Step Trials	Sit Trials
Natural	248/255 – 97.25%	41/42 – 97.62%	27/29 – 93.10%
Momentum-Transfer	257/263 – 97.72%	33/40 – 82.50%	17/19 – 84.75%
Quasi-Static	213/232 – 91.81%	36/39 – 92.31%	27/29 – 93.10%
Combined	718/750 – 95.73%	110/121 – 90.91%	71/77 – 92.21%

Table 3.3: The accuracy of Stability Basins generated using the Input Bounds controller model are reported for cable pull trials at three different force levels of the applied perturbation. The results are aggregated across subjects and the three tested sit-to-stand control strategies.

Cable Pull Force Level	Successful Trials	Step Trials	Sit Trials
Low	170/176 – 96.59%	15/15 – 100%	4/5 – 80.00%
Medium	125/132 – 94.70%	34/38 – 89.47%	24/27 – 88.89%
High	80/85 – 94.12%	61/68 – 89.71%	43/45 – 95.56%

these false successes, we measured the maximum Euclidean distance of each unsuccessful trajectory to its nearest successful trajectory. The false successes are 34.24% closer to their nearest successful trajectories than are the true failure trajectories. Additionally, the onset of failure with

Table 3.4: The accuracy of the Stability Basins formed using the Input Bounds controller model were compared to Stability Basins formed using other methods. The results of the comparison are reported below, and are aggregated across subjects and sit-to-stand strategies. Supplementary material that reports these results for individual subjects and sit-to-stand strategies is available online.

Basin Type	Successful Trials	Step/Sit Trials	False Successful Predictions	False Step/Sit Predictions
LQR	92/750 – 12.27%	197/198 – 99.49%	1/93 – 1.08%	658/855 – 76.96%
FF+FB	351/750 – 46.80%	195/198 – 98.48%	3/354 – 0.85%	399/594 – 67.17%
Input Bounds	718/750 – 95.73%	181/198 – 91.41%	17/735 – 2.31%	32/213 – 15.02%
naive	125/750 – 16.67%	195/198 – 98.48%	3/128 – 2.34%	625/820 – 76.21%

respect to the onset of perturbation for false successes averaged 9.47% earlier than in the true failures. Note that it is plausible that individuals switched control strategies for some subset of the trials (e.g., from Quasi-Static to natural), which could affect the accuracy of the strategy-specific Stability Basins. Because we only characterize trials that are sits or steps as failures, it is possible that an individual switched strategies to successfully stand but would not have reached standing with their original strategy.

We compared the accuracy of Stability Basins computed using the Input Bounds controller to Stability Basins formed using other methods (Tab. 3.4). The Stability Basins formed using all three controllers as well as the naive method for Subject ID 1's natural sit-to-stand strategy are shown in Fig. 3.13.

Stability Basins formed using the LQR controller proposed in previous work [92] correctly predict 12.27% of successful trials. Stability Basins formed using a traditional FF+FB controller correctly predict the outcome of 46.80% of successful trials. Finally, a naive method for estimating stability yields an accuracy of 16.67% for successful trials. Though each of the LQR, FF+FB, and naive methods yield nearly 100% prediction accuracy for trials where failure occurred, the low successful trial prediction rates mean that few trials remain within each Stability Basin, implying a severe underapproximation of the true stable region. We also give the false successful prediction and false failure prediction rates of each method in Tab. 3.4, and display these rates for each strategy in Fig. 3.14. Each of the LQR, FF+FB, and naive approaches for computing Stability Basins has a high false failure prediction rate, further showing that each underestimates the size of the true stable region.

Because the LQR, FF+FB, and naive methods all appear to underestimate the true stable region, we tested the accuracy of the Stability Basins generated by these methods when they are dilated by 5%. This was accomplished by multiplying each of their Stability Basins' zonotope generators by 1.05, as in the target set generation. The classification rates for these methods do improve, though not substantially. As seen in Table 3.5, the successful trial classification rate rises around 3% for the LQR controller, around 6% for the FF+FB controller, and around 5% for the naive method. To

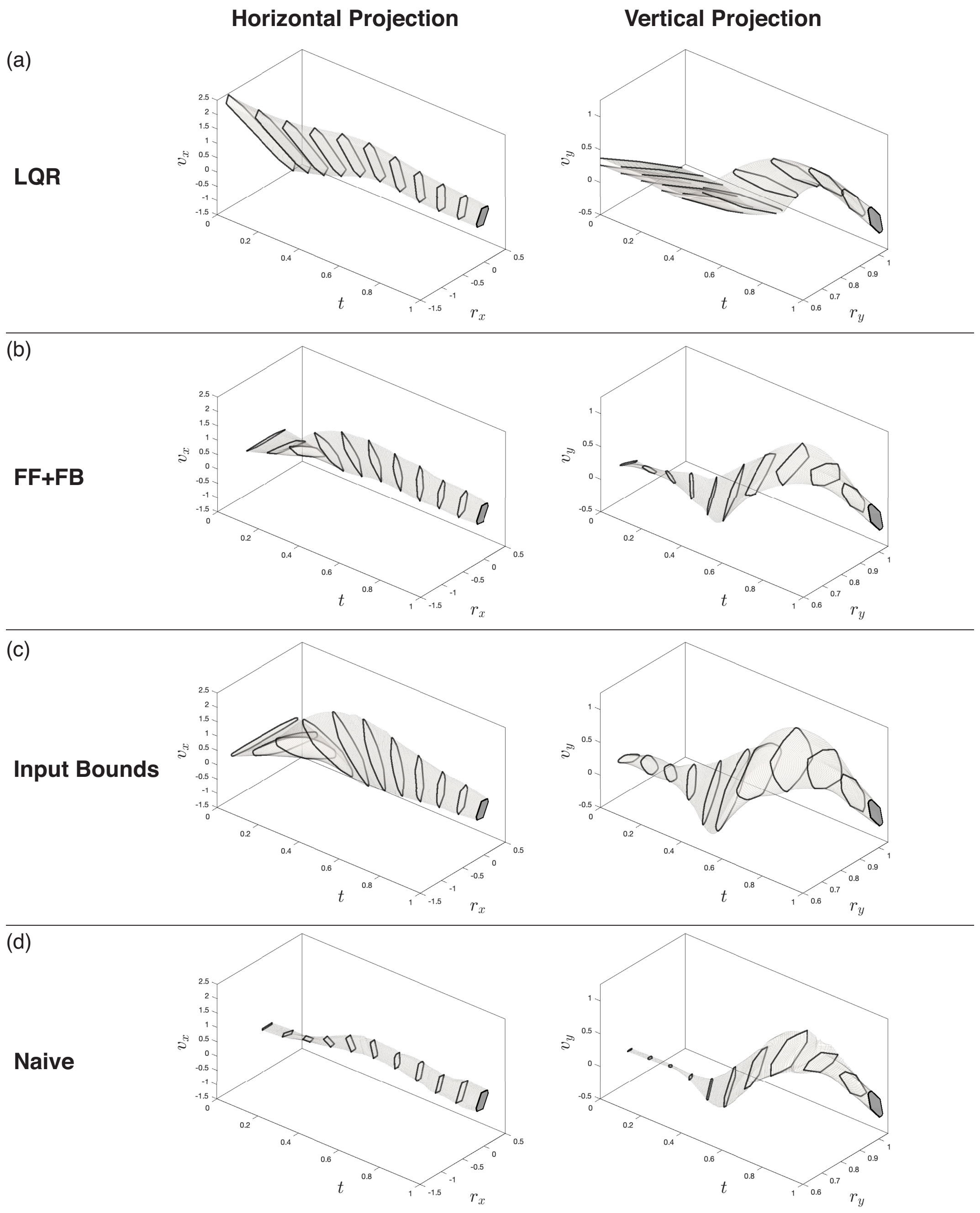


Figure 3.13: The Stability Basins formed for Subject ID 1's natural sit-to-stand strategy using the LQR controller model (Fig. 3.13(a)), FF+FB controller model (Fig. 3.13(b)), Input Bounds controller model (Fig. 3.13(c)), and the naive method (Fig. 3.13(d)) are shown. The horizontal and vertical projections of the Stability Basins are represented as the regions encapsulated by the light grey borders, where black outlines are used to emphasize the changes in shape of cross-sections over time. The projection of the target set X_T is shown as the dark grey region on the right side of each plot.

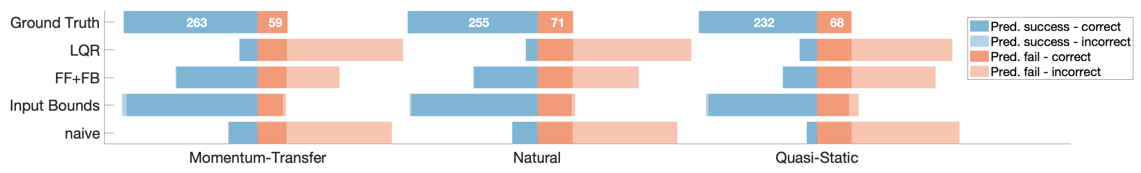


Figure 3.14: This figure compares the predictive accuracy of the SBs formed using the Input Bounds controller to three other methods. Results are presented for each sit-to-stand strategy, aggregated across subjects. The Input Bounds controller's predictions for both successes and failures are near the ground truth, while the three other methods predict many more failures than were experimentally observed.

determine what would happen if we extended this analysis further, we also evaluated the Stability Basins when the basins are dilated by 25%. This further improves the successful trial classification rate of the LQR, FF+FB, and naive methods. However, at this dilation level, the FF+FB has a similar unsuccessful trial classification rate as the Input Bounds controller (89.41% compared to 91.41%), but a 41% higher false step/sit prediction rate (56.83% compared to 15.02%).

Table 3.5: The accuracy of the Stability Basins for each method are compared when dilating the LQR, FF+FB, and naive Stability Basins. The results are aggregated across subjects and STS strategies.

Basin Type	Successful Trials	Step/Sit Trials	False Successful Pred.	False Step/Sit Pred.
Input Bounds	718/750 – 95.73%	181/198 – 91.41%	17/735 – 2.31%	32/213 – 15.02%
LQR (no dilation)	92/750 – 12.27%	197/198 – 99.49%	1/93 – 1.08%	658/855 – 76.96%
FF+FB (no dilation)	351/750 – 46.80%	195/198 – 98.48%	3/354 – 0.85%	399/594 – 67.17%
naive (no dilation)	125/750 – 16.67%	195/198 – 98.48%	3/128 – 2.34%	625/820 – 76.21%
LQR (5% dilation)	113/750 – 15.07%	194/198 – 97.98%	4/117 – 3.42%	637/831 – 76.65%
FF+FB (5% dilation)	390/750 – 52.00%	192/198 – 96.97%	6/396 – 1.52%	360/552 – 65.22%
naive (5% dilation)	159/750 – 21.20%	195/198 – 98.48%	3/162 – 1.85%	591/786 – 75.19%
LQR (25% dilation)	160/750 – 21.33%	188/198 – 94.95%	10/170 – 5.88%	590/778 – 75.84%
FF+FB (25% dilation)	517/750 – 68.93%	177/198 – 89.39%	21/538 – 3.90%	233/410 – 56.83%
naive (25% dilation)	292/750 – 38.93%	189/198 – 95.45%	9/301 – 2.99%	458/647 – 70.79%

This implies that the Input Bounds controller still yields the most accurate estimate of stability. The Input Bounds controller maintains a high accuracy for successful trials, while minimizing incorrect step and sit predictions. It requires no dilation to achieve this performance. The higher false step/sit prediction rates for the LQR and FF+FB controllers despite dilation indicates that the shape of the Stability Basin constructed using the Input Bounds controller is a better approximation of the true stable region.

3.8 Discussion

In theory, stability can be directly characterized by perturbing a subject in an infinite number of ways and drawing a volume around the observed successful trials. Because this approach is practically infeasible, several model-based metrics have been developed for assessing dynamic stability. Many, such as Floquet Multipliers and Lyapunov Exponents, are limited to analyzing small perturbations during periodic motion. Only a few metrics assess the largest perturbations that can be withstood during motion [146], or the stability of aperiodic motion. In this area, Fujimoto and Chou [165] constructed estimates of the stable region for sit-to-stand at a single time based on the extrapolated center of mass concept. We build on these methods by leveraging novel data-driven controller models and reachability analysis to estimate stability over the course of the sit-to-stand motion. By performing a perturbative experiment, we demonstrate that our Stability Basin-based approach accurately predicts stability across a diverse set of body morphologies, perturbations, and sit-to-stand control strategies.

The accuracy, computational-tractability, individualization, and time-varying aspects of the Stability Basin approach make it well suited for widespread deployment and integration with existing clinical methods. This can enable longitudinal studies examining how stability changes over time, and whether a smaller Stability Basin is correlated with fall risk, to be performed both accurately and cheaply. Although we make use of perturbative data for computing Stability Basins here, one advantage of our proposed method is that it only relies on data from successful trials, and does not require a subject to be perturbed to failure. Thus, the Stability Basin method shows promise for characterizing the stability of subjects already at a high risk of falling or injury, such as elderly or rehabilitating subjects. Furthermore, stability throughout the duration of a task can be studied by measuring the size and location of cross-sections of the Stability Basin at particular times. For example, cross-sections of the Stability Basin that examine COM position at seat-off reveal the viable starting postures for each control strategy [92, 165]. We propose that in longitudinal studies with periodic data collection, sudden changes in Stability Basin volume overall or in a portion of the movement can indicate how injury or age contribute to instability.

Our data-driven controller models consist of a feedforward ("predictive") and a linear feedback ("reactive") component. Although there is evidence that humans employ feedforward and feedback terms for control [160], the exact form of these controllers remains unclear. Linear feedback applied about the COM has been sufficient to explain observed walking [166] and standing [167] behavior. However, we found the LQR controller model proposed by Shia et al. [92] as well as the classical FF+FB controller model had poor prediction rates for successful trials, implying that they both underestimate the true stable region of movement. In contrast, our Input Bounds controller model draws from a set of bounded inputs, plus a linear feedback term, that encapsulates inputs from observed successful trials, resulting in improved performance over models that return a single input.

The poor prediction rates of the LQR and FF+FB controller models for successful trials may be explained by considering the inputs that cause trajectories to exit the Stability Basin. A trajectory's input not following a given controller model is not a sufficient condition for the trajectory to exit the Stability Basin generated for that controller model. However, it is a necessary condition: if a trajectory exits the Stability Basin, then the trajectory's input must not have exactly followed the controller model at some point. Therefore, the accuracy of the Stability Basins strongly depends on how well each controller model fits the observed data. As can be seen in Fig. 3.6, the Input Bounds controller accounts for a wider range of inputs than the LQR and FF+FB controllers, and is able to encapsulate the variability present in the observed data. This in turn creates a Stability Basin which better matches the size and shape of the true stable region. This improvement in the accuracy of stability estimation is an important step towards clinical deployment.

In addition to increased accuracy, the computationally-tractable, individualized, and time-varying aspects of the Stability Basin approach make it well suited for widespread deployment and integration with existing clinical methods. Assistive robots may provide an ideal platform for such deployment. The rapid and automated computation of Stability Basins from kinematics alone minimizes the temporal and financial investment in performing this analysis, especially if coupled with computer vision techniques for pose estimation from digital cameras as described in Chap. 2. This can enable longitudinal studies examining how stability changes over time, and whether a smaller Stability Basin is correlated with fall risk, to be performed both accurately and cheaply. Although we make use of perturbative data for computing Stability Basins here, one advantage of our proposed method is that it only relies on data from successful trials, and does not require a subject to be perturbed to failure. Thus, the Stability Basin method shows promise for characterizing the stability of subjects already at a high risk of falling or injury, such as elderly or rehabilitating subjects. Additionally, the TIPM and Input Bounds models are computed for each individual, enabling examination of a wide range of body morphologies and sit-to-stand control strategies. Furthermore, stability throughout the duration of a task can be studied by measuring the size and location of cross-sections of the Stability Basin at particular times. For example, cross-sections of the Stability Basin that examine COM position at seat-off reveal the viable starting postures for each control strategy [92, 165]. In particular, Quasi-Static Stability Basins were previously shown to include COM positions near one's feet at seat-off, whereas Momentum-Transfer Stability Basins can extend much further. We propose that in longitudinal studies with periodic data collection, sudden changes in Stability Basin volume overall or in a portion of the movement can indicate how injury or age contribute to instability. By combining the Stability Basin approach with other clinical techniques, such as electromyography, clinicians can identify muscles associated with unstable portions of motion to target preventative and rehabilitative care. Such detailed analyses can provide deeper insight into the mechanisms underlying motor control.

Our approach can be used in real time to inform controller design for wearable robotic devices because it provides time-varying boundaries on the region inside which an individual is stable. Determining the state at which a robotic system is unable to recover from a perturbation is key to designing dynamically stable controllers [168, 169, 170]. If a prosthetic device or exoskeleton detects that its user has exited the Stability Basin and become unstable, it can adapt its control scheme to aid in recovery. Furthermore, just as current tuning of controllers for prosthetic devices and exoskeletons focuses on minimizing metabolic cost, [171, 172], wearable robotic controllers can be optimized to increase stability by maximizing the volume of the Stability Basins for the combined robot-human system. Thus, the high time-varying accuracy and low investment in computation demonstrate that the Stability Basin approach has the potential to increase access of personalized preventative and rehabilitative mobility care and inform the design of assistive robotic systems for stable movement.

CHAPTER 4

Planning Collision-free Manipulator Trajectories in Real Time using Reachable Sets

Much of this chapter originally appeared in [94]:

> **Patrick Holmes**, Shreyas Kousik, Bohao Zhang, Daphna Raz, Corina Barbalata, Matthew Johnson-Roberson, and Ram Vasudevan. "Reachable Sets for Safe, Real-Time Manipulator Trajectory Design". *Robotics: Science and Systems*, (July 2020).

which extends our work in [95]:

> Shreyas Kousik, **Patrick Holmes**, and Ram Vasudevan. "Safe, aggressive quadrotor flight via reachability-based trajectory design". *ASME Dynamic Systems and Controls Conference*, (October 2019), **Best Student Paper Award**.

For demonstrations of ARMTD, please see our video: `https://youtu.be/ySnux2owlAA`. *Open source code for ARMTD is available here:* `https://github.com/roahmlab/arm_planning`. *A* 5*-minute overview of ARMTD is here:* `https://youtu.be/6tjnh1Yxr_Q`.

4.1 Introduction

Manipulation is a prerequisite function necessary for assistive robots to support older persons and persons with disabilities with many activities of daily living and instrumental activities of daily living. These include tasks like cooking, feeding, grasping and moving objects, opening doors and dressing. Industrial manipulator robots, such as those used in manufacturing, can be extremely heavy and stiff. Collisions between these robots and humans can cause severe injury and death [173]. Smaller, more lightweight robots designed to operate in closer proximity to people, such as the Fetch Mobile Manipulator [41], (Fig. 4.1) can still weigh hundreds of pounds. Collisions with these robots are dangerous, and should be avoided.

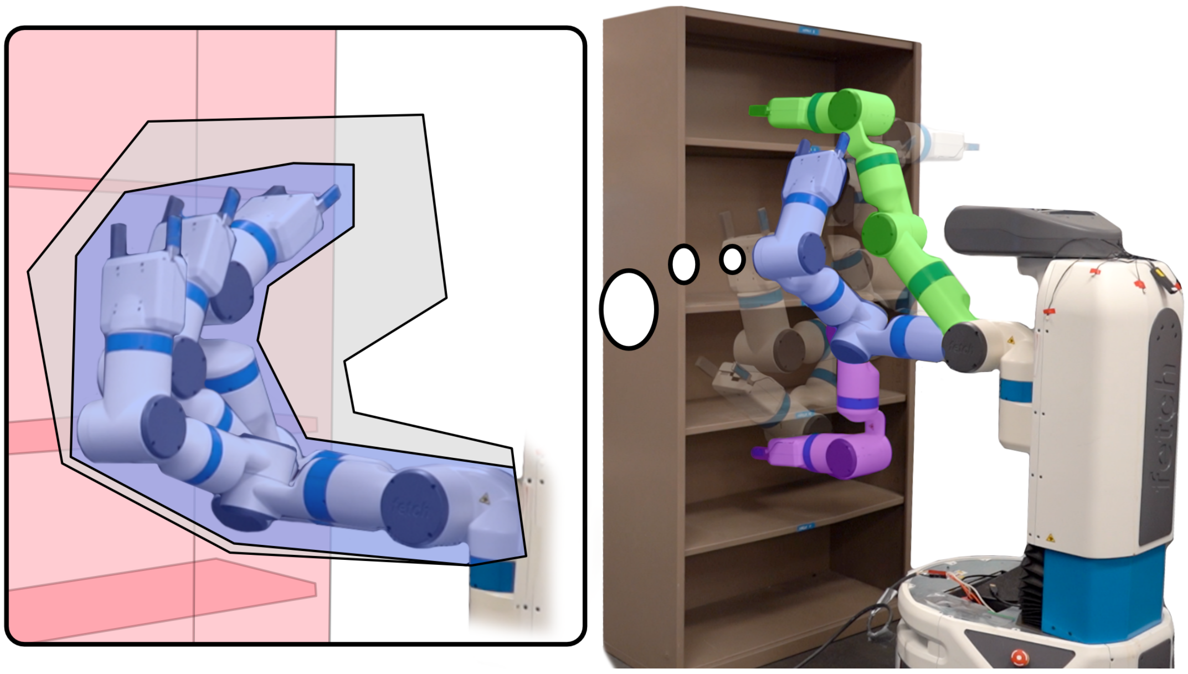


Figure 4.1: ARMTD performs safe, real-time receding-horizon planning for a Fetch arm around a cabinet in real time, from a start pose (purple, low shelf) to a goal (green, high shelf). Several intermediate poses are shown (transparent). The callout on the left, corresponding to the blue intermediate pose, shows a single planning iteration, with the shelf in light red. In grey is the arm's reachable set for a continuum of parameterized trajectories over a short time horizon. The smaller blue set is the subset of the reachable set corresponding to the particular trajectory that was selected for this planning iteration, which is guaranteed not to collide with the obstacle. Over many such trials in simulation and on hardware, ARMTD never crashed. See our video: youtu.be/ySnux2owlAA.

To maximize their utility in long-term care environments, robotic arms should plan collision-free motions in real time. Such performance requires sensing and reacting to the environment as the robot plans and executes motions; in other words, it must perform *receding-horizon planning*, where it iteratively generates a plan while executing a previous plan. Receding-horizon planning is especially important for operating near people since a person may be constantly moving. Manipulation plans should be updated frequently to account for updated predictions of human motion. This chapter addresses **guaranteed-safe receding-horizon trajectory planning for robotic arms**. We call the method Autonomous Reachability-based Manipulator Trajectory Design, or **ARMTD**, introduced in Fig. 4.1.

4.1.1 Related Work

Motion planning can be broadly split into three paradigms, depending on whether safety is enforced by (1) a path planner, (2) a trajectory planner, or (3) a tracking controller.

The first paradigm is commonly used for robotic arm planning, wherein the path planner is responsible for safety. One generates a collision-free path, then smooths it and parameterizes it by time (i.e., converts it into a trajectory) [174, 175]. Such methods often have a tradeoff between safety and real-time performance because they represent paths with discrete points in configuration space [176, 91]. Ensuring safety requires approximations such as buffering the volume of the arm at each discrete point to account for the discretization, or computing the swept volume along the path assuming, e.g., straight lines between points [84]. If one treats the path as a decision variable in a nonlinear optimization program, the gradient of the distance between the arm's volume and obstacles may "push" each configuration out of collision [2, 85, 177]. This means the output path can be treated directly as a trajectory, if the optimization uses path smoothness as the cost. However, this relies on several approximations to achieve real-time performance: finite differencing to bound joint speeds and accelerations, collision penalties in the cost instead of hard constraints, and finite differencing [2] or linearization [85] for the collision-avoidance penalty gradient. This necessitates finer discretization to faithfully represent the robot's kinematics. To enable real-time performance without gradients, one can compute many paths offline, then collision-check at runtime [178, 179]; but for arbitrary tasks, it can be unclear how many paths are necessary, or how to ensure safety if the arm's volume changes (e.g., by grasping an object). Another approach to real-time performance is to plan iteratively in a receding-horizon either by gradient descent (with the same drawbacks as above) [177] or assuming the underlying path planner is safe [180]. In summary, in this paradigm, one must discretize finely, or buffer by a large amount, to achieve safety at the expense of performance.

In the second paradigm, the path planner generates a (potentially unsafe) path, then the trajectory planner attempts to track the path as closely as possible while maintaining safety. In this paradigm, one computes a *reachable set* (RS) for a family of trajectories instead of computing a swept volume for a path. Methods in this paradigm can achieve both safety and real-time performance in receding-horizon planning by leveraging sums-of-squares programming [90, 89, 181, 182, 183, 184] or zonotope reachability analysis [95]. Unfortunately, the methods in this paradigm suffer from the curse of dimensionality, preventing their use with the high-dimensional models of typical arms.

In the third paradigm, one attempts to ensure safety via the tracking controller, instead of in a path or trajectory. Here, one builds a supervisory safety controller for pre-specified trajectories [185] or a set of safe states [186]. Another approach is to compute a safety buffer and associated controller using Hamilton-Jacobi reachability analysis [187, 188], but the curse of dimensionality has prevented applying this to arms.

Reachability-based Trajectory Design (RTD) [89, 189] is a guaranteed-safe receding-horizon trajectory planner that falls into the second paradigm described above. RTD was previously limited to simple 2D planning models, but recently we extended RTD for use in 3D quadrotor flight by leveraging zonotope-based reachable sets [95]. The current chapter further builds on the RTD framework by creating *parameterized reachable sets of arm volumes* and demonstrating their use for designing safe trajectories of manipulators.

To the best of our knowledge, RSs in manipulator planning have only been used for either collision-checking a single, precomputed trajectory [185, 190], or for controlling to a predefined setpoint [191]. In contrast, our proposed ARMTD method generates RSs for a continuum of trajectories, allowing optimization over sets of safe trajectories. Computing such RSs directly is challenging because of the high-dimensional configuration space and nonlinear transformation to workspace used for a typical arm [89, 188].

4.1.2 ARMTD Summary

Two challenges make it difficult to compute reachable sets of manipulator trajectories in workspace. First, the reachable sets depend on the initial condition (angles and velocities) of each joint. This makes it extremely difficult to fully precompute reachable sets of the system, since the many degrees of freedom of the manipulator makes the set of possible initial conditions intractably large. Second, we want to map sets of trajectories in *configuration space* to sets of swept volumes in *workspace*. This involves a nonlinear transformation from joint angles to 3D rotations, and the noncommutative composition of these 3D rotations. These operations are well defined for a single point in configuration space, but it is difficult to find a set representation in configuration space that admits a tractable set-based version of these operations.

Our ARMTD method overcomes these challenges by composing a high-dimensional RS in workspace from low-dimensional **joint reachable sets** (JRSs) of joint configurations. First, ARMTD correctly accounts for initial conditions online by rotating each JRS by the correct initial angle, and only considering subsets of each JRS corresponding to the actual initial velocity of each joint. Second, ARMTD utilizes efficient zonotope-like representations to overapproximate sets of rotation matrices and their products. ARMTD extends the second planning paradigm above by using these RSs to plan safe trajectories in real time. The RS also provides subdifferentiable collision-avoidance, self-intersection, and joint limit constraints for trajectory optimization. Importantly, the RS composition, constraint generation, and gradient evaluation are all parallelizable.

We now provide an overview of ARMTD, also shown in Fig. 4.3. ARMTD begins by specifying a parameterized continuum of kinematic configuration space trajectories, each of which includes a fail-safe maneuver. Offline, ARMTD computes parameterized joint reachable sets, or JRSs, of these trajectories in configuration space. At runtime (in each receding-horizon), it constructs a parameterized RS from the precomputed JRSs. ARMTD intersects the RS with obstacles to generate provably-correct safety constraints. ARMTD then performs trajectory optimization over the parameters, subject to the safety constraints. If it cannot find a feasible solution within a prespecified time limit, the arm continues executing the trajectory from its previous planning iteration (which includes a fail-safe maneuver), guaranteeing perpetual safety [89, 180]. In this work, we only discuss static environments, but this approach can extend to dynamic environments [181] as is necessary for planning around people in a shared workspace.

4.1.3 Contributions

We make the following contributions. First, a method to tractably and conservatively construct the RS of high-dimensional redundant robotic manipulators (Sec. 4.3–4.4). Second, a parallelized method to perform real-time, provably-safe, receding-horizon trajectory optimization (Sec. 4.4). Third, a demonstration in simulation and on hardware, with no collisions (Sec. 4.5 and Supplemental Video), plus a comparison to CHOMP [2]. The remaining sections are Section 4.2 (Arm, Obstacles, and Trajectory Parameters) and Section 4.6 (Conclusion). See our video: `youtu.be/ySnux2owlAA`. Our code is available: `github.com/roahmlab/arm_planning`.

4.2 Arm, Obstacles, and Trajectory Parameters

This section defines notation and representations of an arm and its environment. The goal of this work is to plan collision-free trajectories for a robotic arm operating around obstacles in a receding-horizon framework. We now discuss the arm and its environment, then our receding-horizon framework and parameterized trajectories.

4.2.1 Arm

Consider an arm with $n_q \in \mathbb{N}$ joints (i.e., n_q DOFs) and $n_q + 1$ links, including the 0th link, or *baselink*.

Assumption 4.2.1. *We make the following assumptions/definitions. Each joint is a single-axis revolute joint, attached between a* predecessor *link and a* successor *link. The arm is a single kinematic chain from baselink to end effector; link $i-1$ is joined to link i by joint i for $i = 1, \cdots, n_q$.*

One can create multi-DOF joints using virtual links of zero volume. The *configuration space* is $Q \subseteq \mathbb{S}^{n_q}$, containing *configurations* $q = (q_1, q_2, \cdots, q_{n_q}) \in Q$. The space of joint velocities is $\dot{Q} \subset \mathbb{R}^{n_q}$. There exists a default configuration $0 \in Q$. The *workspace*, $W \subset \mathbb{R}^3$, is the all points in space reachable by any point on the arm in any configuration. The robot's physical limits are as follows. Each joint i has a minimum and maximum position $q^-_{i,\text{lim}}$ and $q^+_{i,\text{lim}}$, maximum absolute speed $\dot{q}_{i,\text{lim}}$ and maximum absolute acceleration $\ddot{q}_{i,\text{lim}}$.

We now describe the kinematic chain. To simplify notation, we let $\prod$ perform right multiplication of matrices with increasing index, e.g.

$$\prod_{i=1}^{3} A_i = A_1 A_2 A_3. \tag{4.1}$$

Each link has a local coordinate frame with the origin located at the link's predecessor joint (the baselink's frame is the global frame). The rotation matrix $R_i(q_i) \in \text{SO}(3)$ describes the rotation of link i relative to link $i-1$ (by joint i). The displacement $l_i \in \mathbb{R}^3$ denotes the position of joint i on link i relative to joint $(i-1)$ in the frame of link i. The set $L_i \subset \mathbb{R}^3$ denotes the volume occupied by the i^{th} link, with respect to its predecessor joint, in the frame of link i. Let $\text{FO}_i : Q \to \mathcal{P}(W)$ give the forward occupancy of link i. That is, the i^{th} link occupies the volume

$$\text{FO}_i(q) \;=\; \left\{ \sum_{j<i} \left(\prod_{n \le j} R_n(q_n)\, l_j \right) \right\} \oplus \left(\prod_{n \le i} R_n(q_n) L_i \right) \subset W. \tag{4.2}$$

Let $\text{FO} : Q \to \mathcal{P}(W)$ give the occupancy of the entire arm: $\text{FO}(q) = \bigcup_{i=1}^{n_q} \text{FO}_i(q)$. Note, the first expression in (4.2) gives the position of joint $(i-1)$ and the second gives the rotated volume of link i.

Example 4.2.2. *Consider an arm with $n_q = 3$. FO_3 as in (4.2) can be written:*

$$\text{FO}_3(q) \;=\; \left\{ R_1(q_1) l_1 + R_1(q_1) R_2(q_2) l_2 \right\} \oplus \left(R_1(q_1) R_2(q_2) R_3(q_3) L_3 \right). \tag{4.3}$$

Notice that in this example, the rotated link volume of the 3rd link is given by $(R_1(q_1)R_2(q_2)R_3(q_3)L_3)$, which is "stacked" on top of the sum of the positions of all predecessor joints.

More generally, for an arm with $n_q > 2$, FO_i as in (4.2) can be written:

$$\begin{aligned} \text{FO}_i(q) \;=\; \Bigg\{ & R_1(q_1) l_1 + R_1(q_1) R_2(q_2) l_2 + \cdots \\ & \cdots + \prod_{j \le (i-1)} R_j(q_j) l_{i-1} \Bigg\} \oplus \left(\prod_{j \le i} R_j(q_j) L_i \right). \end{aligned} \tag{4.4}$$

4.2.2 Obstacles

We denote an *obstacle* as a set $O \subset W$. If the arm's volume at $q \in Q$ is intersecting the obstacle, we say the arm is in *collision*, i.e. $\mathsf{FO}(q) \cap O \neq \emptyset$. We assume the following about obstacles. Each obstacle is compact and static with respect to time (note, one can extend ARMTD to dynamic obstacles [181]). At any time, there are at most $n_{\text{obs}} \in \mathbb{N}$ obstacles in the workspace, and the arm has access to a conservative estimate of the size and location of all such obstacles (we are only concerned with planning, not perception). Let $\mathscr{O} = \{O_1, \cdots, O_{n_O}\}$ denote a set of obstacles.

4.2.3 Receding-Horizon Planning and Timing

ARMTD plans in a receding-horizon way, meaning it generates a short plan, then executes it while generating its next short plan. Every such plan is specified over a compact time interval $T \subset \mathbb{R}$. Without loss of generality (WLOG), since time can be shifted to 0 at the beginning of any plan, we denote $T = [0, t_\text{f}]$. We further specify that ARMTD must generate a new plan every $t_{\text{plan}} < t_\text{f}$ seconds. If a collision-free plan cannot be found within t_{plan} s, the robot must continue the plan from the previous receding-horizon iteration; therefore, we include a fail-safe (braking) maneuver in each plan. The durations t_f and t_{plan} are chosen such that $(t_\text{f} - t_{\text{plan}})$ is large enough for the arm to stop from its maximum joint speeds given its maximum accelerations. This ensures every plan can include a fail-safe maneuver. We abuse notation to let $q : T \to Q$ denote a trajectory plan and $q_i : T \to Q$ denote the trajectory of the i^{th} joint. A plan is *collision-free* if $\mathsf{FO}(q(t)) \cap O = \emptyset \, \forall t \in T, \; \forall \, O \in \mathscr{O}$. Next, we specify the form of each plan.

4.2.4 Trajectory Parameterization

ARMTD plans using parameterized trajectories. We describe the theory, then present our implementation.

4.2.4.1 Theory

Let $K \subset \mathbb{R}^{n_k}$, $n_k \in \mathbb{N}$, be a compact space of *trajectory parameters*, meaning each $k \in K$ maps to a trajectory $q : T \to Q$. We use $q(t; k)$ to denote the configuration parameterized by $k \in K$ at time $t \in T$. So, in each receding-horizon planning iteration, ARMTD attempts to select a single $k \in K$ (via trajectory optimization with obstacles represented as constraints on K).

Definition 4.2.3. *We require $q : T \to Q$ to satisfy three properties for all $k \in K$. First, $q(\cdot\,; k)$ is at least once-differentiable w.r.t. time. Second, $q(0; k) = 0$. Third, $\dot{q}(t_\text{f}; k) = 0$.*

The first property ensures we do not attempt to command a trajectory that includes a jump in joint velocity, which is physically impossible for the arm to follow. The second property uses the fact that all joints are revolute, so $q(0; k) = 0$ without loss of generality. The third property guarantees each parameterized trajectory includes a fail-safe braking maneuver.

Note, the parameterized trajectories are kinematic, not dynamic. We formalize this in the following assumption:

Assumption 4.2.4. *In this work, we assume the arm is capable of perfectly tracking the trajectories specified by the trajectory parameterization.*

In Sec. 4.4.2.2, we ensure trajectories start at the correct initial positions and velocities, and by definition, the trajectories under consideration have no discontinuities in velocity. Furthermore, we obey joint limits in Sec. 4.4.3. Despite these considerations, a robot's built-in controller may not perfectly follow the desired trajectories, and it is possible that desired trajectories would require torques which exceed motor limits.

However, this assumption is common in motion planning [2, 85, 177, 178, 179], because existing controllers can track such trajectories closely (e.g., within 0.01 rad for revolute joints [192, 193]) in the absence of disturbances such as collisions. We find these trajectories sufficient to avoid collision in real-world hardware demonstrations (Sec. 4.5). Also, methods exist for quantifying tracking error [193, 95] and accounting for it at runtime [89, 181].

4.2.4.2 Implementation

We choose a parameterization that is simple yet sufficient for safe planning in arbitrary scenarios (see Sec. 4.5). We define a *velocity parameter* $k^{\mathrm{v}} \in \mathbb{R}^{n_q}$ for the initial velocity $\dot{\tilde{q}}$, and an *acceleration parameter* $k^{\mathrm{a}} \in \mathbb{R}^{n_q}$ that specifies a constant acceleration over $[0, t_{\text{plan}})$. We write $k^{\mathrm{v}} = (k^{\mathrm{v}}_1, \cdots, k^{\mathrm{v}}_{n_q})$ and similarly for k^{a}. We denote $k = (k^{\mathrm{v}}, k^{\mathrm{a}}) \in K \subset \mathbb{R}^{n_k}$, where $n_k = 2n_q$. The trajectories are given by

$$\dot{q}(t;k) = \begin{cases} k^{\mathrm{v}} + k^{\mathrm{a}} t, & t \in [0, t_{\text{plan}}) \\ \frac{k^{\mathrm{v}} + k^{\mathrm{a}} t_{\text{plan}}}{t_{\mathrm{f}} - t_{\text{plan}}}(t_{\mathrm{f}} - t), & t \in [t_{\text{plan}}, t_{\mathrm{f}}], \end{cases}, \tag{4.5}$$

with $q_i(0; k) = 0$ for all k. These trajectories brake to a stop over $[t_{\text{plan}}, t_{\mathrm{f}}]$ with constant acceleration. An example trajectory of $\dot{q}_i(t; k)$ is shown in Fig. 4.2.

We require that K is compact to perform reachability analysis (Sec. 4.3). Let K_i denote the parameters for joint i. For each joint i, we specify $K_i = K^{\mathrm{v}}_i \times K^{\mathrm{a}}_i$, where

$$K^{\mathrm{v}}_i = \left[\overline{k^{\mathrm{v}}_i} - \Delta k^{\mathrm{v}}_i,\ \overline{k^{\mathrm{v}}_i} + \Delta k^{\mathrm{v}}_i\right], \quad K^{\mathrm{a}}_i = \left[\overline{k^{\mathrm{a}}_i} - \Delta k^{\mathrm{a}}_i,\ \overline{k^{\mathrm{a}}_i} + \Delta k^{\mathrm{a}}_i\right], \tag{4.6}$$

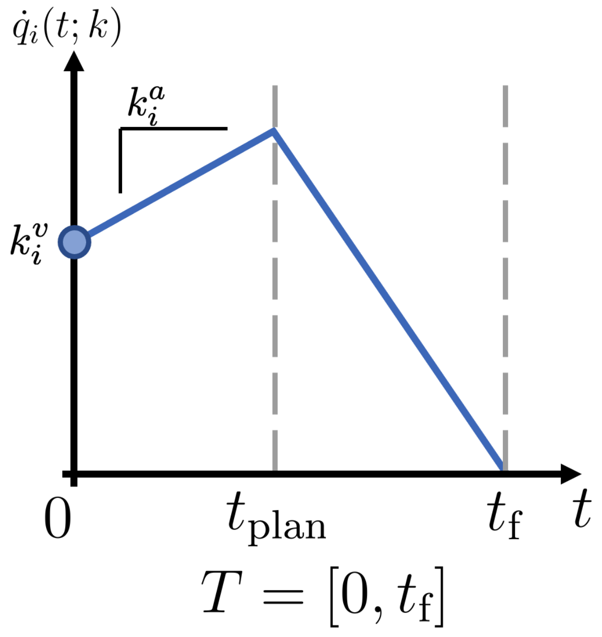


Figure 4.2: An example joint velocity trajectory. The velocity parameter k_i^v specifies the initial velocity of the joint, while the acceleration parameter k_i^a specifies a constant acceleration until t_{plan}. Between t_{plan} and t_{f}, the velocity decreases to zero at a constant rate.

with $\overline{k_i^{\text{v}}}$, $\overline{k_i^{\text{a}}}$, Δk_i^{v}, $\Delta k_i^{\text{a}} \in \mathbb{R}$ and $\Delta k_i^{\text{v}}, \Delta k_i^{\text{a}} \geq 0$. To implement acceleration limits (i.e., to bound K_i^{a}), we ensure

$$K_i^{\text{a}} = \left[\max\left\{-\ddot{q}_{i,\text{lim}}, \overline{k_i^{\text{a}}} - \Delta k_i^{\text{a}}\right\}, \min\left\{\ddot{q}_{i,\text{lim}}, \overline{k_i^{\text{a}}} + \Delta k_i^{\text{a}}\right\}\right]. \tag{4.7}$$

Next, we use these parameterized trajectories to build parameterized reachable sets of joint configurations.

4.3 Offline Reachability Analysis

ARMTD uses short parameterized trajectories of joint angles for trajectory planning. We now describe a **Joint Reachable Set** (JRS) containing all such parameterized trajectories. All computations in this section are performed offline.

4.3.1 Joint Reachable Sets

4.3.1.1 Theory

Since each q_i represents a rotation, we examine trajectories of $\cos(q_i)$ and $\sin(q_i)$, as shown in Fig. 4.3. By Def. 4.2.3, $q(\cdot\,; k)$ is at least once differentiable. We can write a differential equation of the sine and cosine as a function of the joint trajectory, where k is a constant:

$$\frac{d}{dt}\begin{bmatrix}\cos(q_i(t;k))\\ \sin(q_i(t;k))\\ k\end{bmatrix} = \begin{bmatrix}-\sin(q_i(t;k))\dot{q}_i(t;k)\\ \cos(q_i(t;k))\dot{q}_i(t;k)\\ 0\end{bmatrix}. \tag{4.8}$$

We then define the parameterized JRS of the i^{th} joint:

$$\begin{aligned}\mathscr{J}_i = \Big\{&(c,s,k) \in \mathbb{R}^2 \times K \mid \exists\, t \in T \text{ s.t. } q_i \text{ as in (4.5)},\\ &c = \cos(q_i(t;k)),\ s = \sin(q_i(t;k)),\\ &\text{and } \tfrac{d}{dt}\big(\cos(q_i(t;k)), \sin(q_i(t;k)), k\big) \text{ as in (4.8)}\Big\}.\end{aligned} \tag{4.9}$$

We account for different initial joint angles, and use the JRSs to overapproximate the forward occupancy FO, in Sec. 4.4.

4.3.1.2 Implementation

We represent (4.9) using zonotopes, introduced in Chap. 1.5.2. As a refresher, a *zonotope* is a set in $\mathbb{R}^n$ in which each element is a linear combination of a *center* $x \in \mathbb{R}^n$ and *generators* $g^1, \cdots, g^p \in \mathbb{R}^n,\ p \in \mathbb{N}$:

$$Z = \left\{ y \in \mathbb{R}^n \;\middle|\; y = x + \sum_{i=1}^{p} \beta^i g^i,\ -1 \le \beta^i \le 1 \right\}. \tag{4.10}$$

Throughout this section, we introduce the following notation. Let Greek lowercase letters in angle brackets be indeterminate variables (e.g., $\langle\sigma\rangle$). We denote $Z = (x, g^i, \langle\beta^i\rangle)^p$ as shorthand for a zonotope with center x, a set of generators $\{g^i\}_{i=1}^p$, and a set of indeterminate coefficients $\{\langle\beta^i\rangle\}_{i=1}^p$ corresponding to each generator. When an indeterminate coefficient $\langle\beta^i\rangle$ is *evaluated*, or assigned a particular value, we write β^i (i.e., without angle brackets).

To represent the JRS, we first choose a time step $\Delta t \in \mathbb{R}$ such that $\frac{t_\text{f}}{\Delta t} \in \mathbb{N}$ and partition T into $\frac{t_\text{f}}{\Delta t}$ closed intervals each of length Δt, indexed by $\mathbb{N}_T = \left\{0, 1, \cdots, \frac{t_f}{\Delta t} - 1\right\}$. We represent $\mathscr{J}_i$ using one zonotope per time interval, which is returned by $J_i : \mathbb{N}_T \to \mathcal{P}(\mathbb{R}^2 \times K)$. For example, the zonotope $J_i(n)$ corresponds to the n-th time interval $[n\Delta t, (n+1)\Delta t]$. We abuse notation and let t index the subinterval of T that contains it, so that $J_i(t) = J_i\left(\lfloor t/\Delta t \rfloor\right)$ where $\lfloor \cdot \rfloor$ rounds down to the nearest integer. We use similar notation for the center, generators, and indeterminates.

To compute $\mathscr{J}_i$, we first make an initial condition zonotope $J_i(0) \subset \mathbb{R}^2 \times K$.

$$J_i(0) = \left(\tilde{x}_i, \{\tilde{g}_i^{\mathrm{v}}, \tilde{g}_i^{\mathrm{a}}\}, \{\langle \tilde{\kappa}_i^{\mathrm{v}} \rangle, \langle \tilde{\kappa}_i^{\mathrm{a}} \rangle\} \right), \tag{4.11}$$

with $\tilde{x}_i = [1, 0, \overline{k_i^{\mathrm{v}}}, \overline{k_i^{\mathrm{a}}}]^\top$, $\tilde{g}_i^{\mathrm{v}} = [0, 0, \Delta k_i^{\mathrm{v}}, 0]^\top$, $\tilde{g}_i^{\mathrm{a}} = [0, 0, 0, \Delta k_i^{\mathrm{a}}]^\top$. The indeterminates $\langle \tilde{\kappa}_i^{\mathrm{v}} \rangle$ and $\langle \tilde{\kappa}_i^{\mathrm{a}} \rangle$ correspond to $\tilde{g}_i^{\mathrm{v}}$ and $\tilde{g}_i^{\mathrm{a}}$. $J_i(0)$ contains K_i^{v} and K_i^{a} in the k_i^{v} and k_i^{a} dimensions.

Finally, we use the open-source toolbox CORA [102] with the time partition, differential equation (4.8) and (4.5), and initial set $J_i(0)$ to overapproximate (4.9). CORA is introduced in Chap. 1.5.2. Importantly, by [194, Thm. 3.3 and Prop. 3.7], one can prove the following:

$$\mathscr{J}_i \subseteq \bigcup_{t \in T} J_i(t). \tag{4.12}$$

JRSs are illustrated in Fig. 4.3.

Next, we note a property of the zonotope JRS:

Lemma 4.3.1. *There exist $J_i : \mathbb{N}_T \to \mathcal{P}(\mathbb{R}^2 \times K)$ that overapproximate $\mathscr{J}_i$ as in* (4.12) *such that, for each $t \in T$, $J_i(t)$ has only one generator with a nonzero element, equal to Δk_i^{v}, in the dimension corresponding to k_i^{v}; we denote this generator $g_i^{\mathrm{v}}(t)$. Similarly, $J_i(t)$ has only one generator $g_i^{\mathrm{a}}(t)$ (distinct from $g_i^{\mathrm{v}}(t)$) with a nonzero element, Δk_i^{a}, for k_i^{a}.*

Proof. Given $J_i(0)$, the subsequent zonotope $J_i(\Delta t)$ is computed as $J_i(\Delta t) = e^{F\Delta T} J_i(0) + E$, where F is found by linearizing the dynamics (4.8) at $t = 0$ and E is a set that overapproximates the linearization error and the states reached over the interval $[0, \Delta t]$ [194, Section 3.4.1]. This linearized forward-integration and error-bounding procedure is applied to $J_i(\Delta t)$ to produce $J_i(2\Delta t)$, and so on, to compute all $J_i(t)$ in (4.12). Since $\dot{k} = 0$, we have that $e^{F\Delta}\tilde{g}_i^{\mathrm{v}}$ equals $\tilde{g}_i^{\mathrm{v}}$ in the k dimensions (and therefore each $g_i^{\mathrm{v}}(t)$ does as well, and similarly for $g_i^{\mathrm{a}}(t)$). Since the zero dynamics have no linearization error, one can define E to have zero volume in the k dimensions [194, Proposition 3.7], meaning no generator of any $J_i(t)$ has a nonzero element in the k dimensions, except for $g_i^{\mathrm{v}}(t)$ and $g_i^{\mathrm{a}}(t)$ (which are defined with such nonzero elements). ■

Note, the zonotopes created by the open-source toolbox [102] satisfy Lem. 4.3.1. For each $J_i(t)$, we denote the center $x_i(t)$, the generators $\{g_i^{\mathrm{v}}(t), g_i^{\mathrm{a}}(t), g_i^j(t)\}$, and the corresponding indeterminates $\left\{\langle \kappa_i^{\mathrm{v}}(t) \rangle, \langle \kappa_i^{\mathrm{a}}(t) \rangle, \langle \beta_i^j(t) \rangle\right\}$ for $j = 1, \cdots, p(t) \in \mathbb{N}$. We write $p(t)$ since the number of generators is not necessarily the same for each $J_i(t)$ [102]. For all t except 0, $g_i^{\mathrm{v}}(t)$ and $g_i^{\mathrm{a}}(t)$ may have nonzero elements in the cosine and sine dimensions, due to nonzero dynamics and linearization error. The generators $g_i^{\mathrm{v}}(t)$ and $g_i^{\mathrm{a}}(t)$ are important because they let us obtain a subset of the JRS corresponding to a particular choice of parameters k_i^{v} and k_i^{a}. This fact will be important in the next section, where use the JRSs online to build an RS for the arm and identify unsafe plans in each receding-horizon iteration.

4.4 Online Planning

We now present ARMTD's online algorithm for a single receding-horizon iteration (see Alg. 3 and Fig. 4.3). First, we construct the parameterized RS of the entire arm from the JRS of each joint. Second, we identify unsafe trajectory plans which may cause collision with an obstacle or self intersection. Third, we optimize over the safe plans to minimize an arbitrary cost function. If no solution is found, we execute the previous plan's fail-safe maneuver. Before diving in, we present some concepts which are necessary for understanding the online planning method.

4.4.1 Matrix Zonotopes, Rotatotopes, and Slicing

First, we introduce **matrix zonotopes**, which are zonotopes whose elements are matrices. The forward occupancy map FO uses rotation matrices formed from the cosine and sine of each joint. By overapproximating sets of these rotation matrices using matrix zonotopes, we will overapproximate FO in Sec. 4.4.2.2. Note that matrices form a vector space, so matrix zonotopes are analagous to zonotopes.

Definition 4.4.1. *A* matrix zonotope $M \subset \mathbb{R}^{n\times n}$ *is a set of matrices parameterized by a center X and generators $G^1, \cdots, G^m$:*

$$M = \left\{ A \in \mathbb{R}^{n\times n} \;\middle|\; A = X + \sum_{j=1}^{m} G^j \lambda^j, -1 \le \lambda^j \le 1 \right\}. \tag{4.13}$$

We use $M = (X, G^j, \langle \lambda^j \rangle)^m$ as shorthand for a matrix zonotope with center X, generators $\{G^j\}_{j=1}^m$, and indeterminate coefficients $\{\langle \lambda^j \rangle\}_{j=1}^m$.

Note, superscripts are indices, not exponentiation, of matrix zonotope generators.

Next, we introduce the concept of matrix zonotope multiplication. Just as a matrix of appropriate dimension can be multiplied by another matrix or a vector, a matrix zonotope can be multiplied by another matrix zonotope or a zonotope.

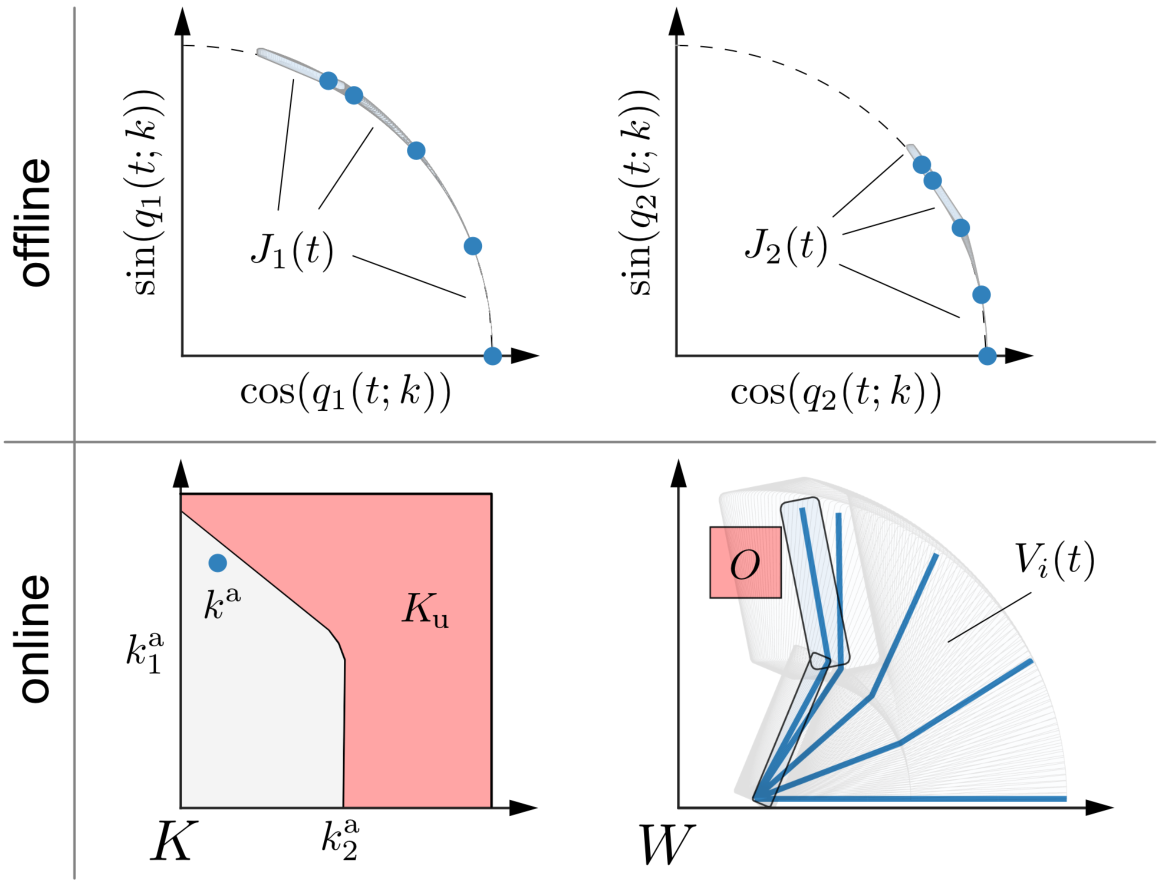


Figure 4.3: An overview of the proposed method for a 2-D, 2-link arm. Offline, ARMTD computes the JRSs, shown as the collection of small grey sets $J_i(t)$ overlaid on the unit circle (dashed) in the sine and cosine spaces of two joint angles. Note that each JRS is conservatively approximated, and parameterized by trajectory parameters K. Online, the JRSs are composed to form the arm's reachable set $V_i(t)$ (large light grey sets in W), maintaining a parameterization by K. The obstacle O (light red) is mapped to the unsafe set of trajectory parameters $K_{\mathrm{u}} \subset K$ on the left, by intersection with each $V_i(t)$. The parameter k^{a} represents a trajectory, shown at five time steps (blue arms in W, and blue dots in joint angle space). The subset of the arm's reachable set corresponding to k^{a} is shown for the last time step (light blue boxes with black border), critically not intersecting the obstacle, which is guaranteed because $k^{\mathrm{a}} \notin K_{\mathrm{u}}$.

Example 4.4.2. *Consider a matrix zonotope $M \subset \mathbb{R}^{n\times n}$, with $M = (X, G, \langle\lambda\rangle)$ and a zonotope $Z \subset \mathbb{R}^n$, with $Z = (x, g, \langle\beta\rangle)$. Their product MZ can be written as*

$$MZ = \left\{ y \in \mathbb{R}^n \;\middle|\; y = Az, A \in M, z \in Z \right\} \tag{4.14}$$
$$= \left\{ y \in \mathbb{R}^n \;\middle|\; y = Az, A = X + G\lambda, z = x + g\beta \right\} \tag{4.15}$$
$$= \left\{ y \in \mathbb{R}^n \;\middle|\; y = Xx + Xg\beta + Gx\lambda + Gg\lambda\beta \right\} \tag{4.16}$$

where β and λ are understood to be within $[-1, 1]$.

The product MZ looks like a zonotope with center Xx and generators $\{Xg, Gx, Gg\}$. However, the indeterminate on Gg in (4.16) is the product of indeterminates $\langle\lambda\rangle$ and $\langle\beta\rangle$. So, MZ looks like a zonotope, but where generators may have more than one indeterminate coefficient. We call such an object a **rotatotope**, because we will use it to represent sets of rotation matrices multiplied by zonotopes in Sec. 4.4.2.2.

More generally, we define rotatotopes as follows:

Definition 4.4.3. *Let $Z = (x, g^i, \langle\beta^i\rangle)^p$ be a zonotope and $M = (X, G^j, \langle\lambda^j\rangle)^m$ be a matrix zonotope. Let $MZ := \{y \in \mathbb{R}^n \mid y = Az,\ A \in M,\ z \in Z\} \subset \mathbb{R}^n$. We call MZ a* rotatotope, *which can be written:*

$$MZ = \left\{ y \in \mathbb{R}^n \ \mid\ y = Xx + \textstyle\sum_i \beta^{(i)} X g^{(i)} + \sum_j \lambda^{(j)} G^{(j)} x + \sum_{i,j} \beta^{(i)} \lambda^{(j)} G^{(j)} g^{(i)} \right\}, \tag{4.17}$$

where $i = 1, \cdots, p$ and $j = 1, \cdots, m$, and each β and λ are understood to be in $[-1, 1]$.
We use the shorthand $MZ = (\hat{x}, \hat{g}^r, \langle\gamma^r\rangle)^s$ where $\hat{x} = Xx$, $s = (p+1)(m+1) - 1$, and the generator and coefficient sets are

$$\{\hat{g}^r\}_{r=1}^s = \{Xg^1, \cdots, Xg^p, G^1x, \cdots, G^mx, G^1g^1, \cdots, G^mg^p\}$$
$$\{\langle\gamma^r\rangle\}_{r=1}^s = \{\langle\beta^1\rangle, \cdots, \langle\beta^p\rangle, \langle\lambda^1\rangle, \cdots, \langle\lambda^m\rangle, \langle\beta^1\lambda^1\rangle, \cdots, \langle\beta^p\lambda^m\rangle\}.$$

Rotatotopes are a special class of polynomial zonotopes [102]. Each $\langle\gamma^r\rangle$ for $r > p + m$ is a product of indeterminate coefficients from M and Z. For a pair of indeterminate coefficients $\langle\gamma^1\rangle$ and $\langle\gamma^2\rangle$, the notation $\langle\gamma^1\gamma^2\rangle$ indicates the product $\langle\gamma^1\rangle\langle\gamma^2\rangle$. We call $\langle\gamma^1\rangle$ and $\langle\gamma^2\rangle$ the factors *of $\langle\gamma^1\gamma^2\rangle$.*

Next, we show that the product of *multiple* matrix zonotopes with a zonotope still yields a rotatotope.

Example 4.4.4. *Consider two matrix zonotopes $M_1 \subset \mathbb{R}^{n\times n}$ and $M_2 \subset \mathbb{R}^{n\times n}$, with $M_1 =$*

$(X_1, G_1, \langle\lambda_1\rangle)$ and $M_2 = (X_2, G_2, \langle\lambda_2\rangle)$. Also let $Z \subset \mathbb{R}^n$, with $Z = (x, g, \langle\beta\rangle)$. Their product $M_1 M_2 Z \subset \mathbb{R}^n$ can be written as

$$\begin{aligned}
M_1 M_2 Z &= \left\{ y \in \mathbb{R}^n \;\middle|\; y = A_1 A_2 z, A_1 \in M_1, A_2 \in M_2, z \in Z \right\} && (4.18)\\
&= \left\{ y \in \mathbb{R}^n \;\middle|\; y = A_1 A_2 z, A_1 = X_1 + G_1\lambda_1, A_2 = X_2 + G_2\lambda_2, z = x + g\beta \right\} && (4.19)\\
&= \left\{ y \in \mathbb{R}^n \;\middle|\; y = X_1X_2x + X_1G_2x\lambda_2 + G_1X_2x\lambda_1 + G_1G_2x\lambda_1\lambda_2 + \ldots \right. && (4.20)\\
&\qquad \left. + \ldots X_1X_2g\beta + X_1G_2g\lambda_2\beta + G_1X_2g\lambda_1\beta + G_1G_2g\lambda_1\lambda_2\beta \right\}.
\end{aligned}$$

So, $M_1M_2Z = (\hat{x}, \hat{g}^r, \langle\gamma^r\rangle)^s$ is a rotatotope with center $\hat{x} = X_1X_2x$,

$$\begin{aligned}
\{\hat{g}^r\}_{r=1}^s &= \{X_1G_2x, G_1X_2x, G_1G_2x, X_1X_2g, X_1G_2g, G_1X_2g, G_1G_2g\},\\
\{\langle\gamma^r\rangle\}_{r=1}^s &= \{\langle\lambda_2\rangle, \langle\lambda_1\rangle, \langle\lambda_1\lambda_2\rangle, \langle\beta\rangle, \langle\lambda_2\beta\rangle, \langle\lambda_1\beta\rangle, \langle\lambda_1\lambda_2\beta\rangle\}.
\end{aligned}$$

We use this fact in Sec. 4.4.2.2 to string together multiplications that overapproximate the forward occupancy map FO.

Two useful properties follow from the rotatotope definition, which we state here:

Lemma 4.4.5. *A matrix zonotope times a rotatotope is a rotatotope.*

Lemma 4.4.6. *(Zono/rotatotope Minkowski sum) Consider two zonotopes $X = (x, g_X^i, \langle\zeta^i\rangle)^n$ and $Y = (y, g_Y^j, \langle\psi^j\rangle)^m$. Then $X \oplus Y = (x + y, \{g_X^i, g_Y^j\}, \{\langle\zeta^i\rangle, \langle\psi^j\rangle\})_{i=1,j=1}^{i=n,j=m}$, which is a zonotope centered at $x + y$ with all the generators and indeterminates of both X and Y. Similarly, for two rotatotopes, $V = (v, g_V^i, \langle\mu^i\rangle)^n$ and $W = (w, g_W^j, \langle\omega^j\rangle)^m$),*

$$V \oplus W = \left(v + w, \{g_V^i, g_W^j\}, \{\langle\mu^i\rangle, \langle\omega^j\rangle\}\right)_{i=1,j=1}^{i=n,j=m}. \quad (4.21)$$

That is, the Minkowski sum of these sets is given by the sum of the centers and the union of the generators/indeterminate sets.

Lastly, we introduce the concept of **slicing**, which is the process of *evaluating* (or "fixing") the values of indeterminate coefficients. In the definitions of zonotopes and rotatotopes, indeterminate coefficients may take any value between -1 and 1. Say we had the indeterminate $\langle\gamma\rangle$, where $\gamma \in [-1, 1]$. If we choose a specific value for γ (e.g. $\gamma = -0.47$), we can remove $\langle\gamma\rangle$ from the list of indeterminates and update the zonotope or rotatotope accordingly. This operation yields a subset of the original zonotope or rotatotope. We define the `slice` operation in Alg. 1.

We slice a zonotope by taking in a set of indeterminate coefficients and corresponding values with which to evaluate them. We evaluate an indeterminate by multiplying its associated generator by the given value. We then *remove* the corresponding indeterminate from the set. Since any

Algorithm 1 $Z_{\text{sliced}} = \texttt{slice}\left(Z, \{\langle\sigma^j\rangle\}_{j=1}^n, \{\sigma^j\}_{j=1}^n\right)$

```
1: // Let $Z = (x, g^i, \langle\beta^i\rangle)^p$ denote the input zonotope or rotatotope
2: $Z_{\text{sliced}} \leftarrow (x, g^i, \langle\beta^i\rangle)^p$ // allocate output
3: for $i = 1, \cdots, p$ // iterate over generator/indeterminate pairs
4:     for $j = 1, \cdots, n$ // iterate over input values
5:         if $\langle\sigma^j\rangle \in \langle\beta^i\rangle$
6:             $g^i \leftarrow \sigma^j g^i$ // multiply generator by value
7:             $\langle\beta^i\rangle \leftarrow \langle\beta^i\rangle \setminus \langle\sigma^j\rangle$ // remove evaluated indeterminate
8:         end if
9:     end for
10:    if $\langle\beta^i\rangle = \emptyset$ // if fully-sliced, then $g^i$ is no longer needed
11:        $x \leftarrow x + g^i$ and $g^i \leftarrow \emptyset$ // shift center, remove generator
12:    end if
13: end for
```

zonotope generator has only one indeterminate, once its indeterminate is evaluated, it is called *fully-sliced*, and added to the center of the zonotope. The process is identical for rotatotopes; however, since rotatotopes may have multiple indeterminates per generator, a generator may be sliced without being fully-sliced.

We now define removing factors generically. We denote the *removal* of the i^{th} indeterminate coefficient of $\langle\gamma^1\gamma^2\cdots\gamma^n\rangle$ as:

$$\langle\gamma^1\gamma^2\cdots\gamma^n\rangle \setminus \langle\gamma^i\rangle = \langle\gamma^1\gamma^2\cdots\gamma^{i-1}\gamma^{i+1}\cdots\gamma^n\rangle. \tag{4.22}$$

We define $\langle\gamma^1\gamma^2\cdots\gamma^n\rangle \setminus \langle\gamma^1\gamma^2\cdots\gamma^n\rangle = \emptyset$. We write $\langle\sigma\rangle \in \langle\gamma^1\gamma^2\cdots\gamma^n\rangle$ to denote that $\langle\sigma\rangle$ is a factor of $\langle\gamma^1\gamma^2\cdots\gamma^n\rangle$.

In the next section, we construct rotatotopes that overapproximate the forward occupancy map of each link by *stacking* rotatotopes representing link volume on top of rotatotopes representing joint positions:

4.4.2 Reachable Set Construction

4.4.2.1 Theory

Recall that ARMTD plans while the robot is executing its previous plan. Therefore, ARMTD must estimate its future initial condition $(\tilde{q}, \dot{\tilde{q}}) \in Q \times \dot{Q}$ as a result of its previous plan by integrating (4.8) for t_{plan} seconds. At the beginning of each online planning iteration, we use $(\tilde{q}, \dot{\tilde{q}})$ to compose the RS of the arm from the low-dimensional JRSs. Denote each link's RS $\mathscr{L}_i$, formed from all $\mathscr{J}_j$

with $j \leq i$:

$$\mathscr{L}_i = \Big\{ \big(Y, k\big) \in \mathcal{P}(W) \times K \;\Big|\; \exists\, t \in T \text{ s.t.} \\ \dot{q}_i(0; k) = \dot{\tilde{q}}_i,\ Y = \text{FO}_i(q(t; k) + \tilde{q}), \\ \text{and } (\cos(q_j(t; k)), \sin(q_j(t; k)), k) \in \mathscr{J}_j\ \forall\, j \leq i \Big\} \tag{4.23}$$

with FO_i as in (4.2). Each $\mathscr{L}_i$ is formed by trajectories which start at the given initial conditions $(\tilde{q}, \dot{\tilde{q}})$. The RS of the entire arm, $\mathscr{L} \subset W \times K$, is then $\mathscr{L} = \bigcup_i \mathscr{L}_i$.

4.4.2.2 Implementation

It is important that we overapproximate $\mathscr{L}$ to guarantee safety when planning. To do this, we overapproximate FO for all configurations in each $\mathscr{J}_i$ (see Alg. 2).

First, we fix $\dot{\tilde{q}}$ by obtaining subsets of the JRSs containing trajectories with the given initial velocity. Recall from Lem. 4.3.1, we write the zonotopes that comprise the JRS as $J_i(t)$, with center $x_i(t)$ and generators $\{g_i^{\text{v}}(t), g_i^{\text{a}}(t), g_i^j(t)\}$, along with their corresponding indeterminates $\big\{\langle\kappa_i^{\text{v}}(t)\rangle, \langle\kappa_i^{\text{a}}(t)\rangle, \langle\beta_i^j(t)\rangle\big\}$ for $j = 1, \cdots, p(t) \in \mathbb{N}$. The generator $g_i^{\text{v}}(t)$ and its indeterminate $\langle\kappa_i^{\text{v}}(t)\rangle$ correspond to the trajectory parameter k_i^{v}; we call these generators k^{v}*-sliceable*. Similarly, the generator $g_i^{\text{a}}(t)$ and its indeterminate $\langle\kappa_i^{\text{a}}(t)\rangle$ correspond to the trajectory parameter k_i^{a}; we call these generators k^{a}*-sliceable*. Essentially, $g_i^{\text{v}}(t)$ encodes a range of $\cos(q_i(t; k))$ and $\sin(q_i(t); k)$ corresponding to a range of initial velocities K_i^{v}. We restrict ourselves to only consider trajectories which start with the correct initial velocity $\dot{\tilde{q}}$ by *slicing* the k^{v}-sliceable generators.

For each joint i, each $J_i(t)$ has generator $g_i^{\text{v}}(t)$, with indeterminate $\langle\kappa_i^{\text{v}}(t)\rangle$ and nonzero element Δk_i^{v} corresponding to the k_i^{v} dimension. Also, $x_i(t)$ (the center of $J_i(t)$) has the value $\overline{k_i^{\text{v}}}$ in that same dimension. We use $\dot{\tilde{q}}$ to slice each $J_i(t)$:

$$S_i(t) = \texttt{slice}\big(J_i(t),\ \langle\kappa_i^{\text{v}}(t)\rangle,\ (\dot{\tilde{q}}_i - \overline{k_i^{\text{v}}})/\Delta k_i^{\text{v}}\big) \tag{4.24}$$

Note, we ensure $\dot{\tilde{q}} \in K_v$ later in this section. We denote

$$S_i(t) = (x_i^{\text{v}}(t), \big\{g_i^{\text{a}}(t), g_i^j(t)\big\}, \big\{\langle\kappa_i^{\text{a}}(t)\rangle, \langle\beta_i^j(t)\rangle\big\})^{p(t)}, \tag{4.25}$$

where $x_i^{\text{v}}(t)$ is the new (shifted) center and $p(t) \in \mathbb{N}$ is the new number of generators, other than $g_i^{\text{a}}(t)$, left after slicing. $S_i(t)$ contains a set of $\cos(q_i(t; k))$ and $\sin(q_i(t; k))$ reachable for a single value of k_i^{v}, but for a range of k_i^{a}. Denote the components of $S_i(t)$ as $x_i^{\text{v}}(t) = [c_i^{\text{v}}, s_i^{\text{v}}, \dot{\tilde{q}}_i, \overline{k_i^{\text{a}}}]^\top$, $g_i^{\text{a}}(t) = [c_i^{\text{a}}, s_i^{\text{a}}, 0, \Delta k_i^{\text{a}}]^\top$ and $g_i^j(t) = [c_i^j, s_i^j, 0, 0]^\top$ for each $j = 1, ..., p(t)$. Note from Lem. 4.3.1

that c_i^{a} and s_i^{a} are generally non-zero, and Δk_i^{a} is constant.

Example 4.4.7. *To understand slicing in* (4.24)*, consider a particular initial velocity $\dot{\tilde{q}}_i = \frac{\pi}{6}$ rad/s. This implies we want the trajectory parameter $k_i^{\mathrm{v}} = \frac{\pi}{6}$. We want to obtain the subset of the JRS representing reachable joint angles corresponding to this particular trajectory parameter.*

For each $J_i(t)$, only the generator $g_i^{\mathrm{v}}(t)$ is non-zero in the dimension corresponding to this trajectory parameter, meaning this k^{v}-sliceable generator is solely responsible for the volume of the reachable set in this dimension. Choosing a particular value of the trajectory parameter means fixing this generator's indeterminate $\langle \kappa_i^{\mathrm{v}}(t) \rangle$ to a particular value. Since the k^{v}-sliceable generator is (generally) non-zero in the cosine and sine dimensions as well, the slicing operation returns a subset of the JRS in those dimensions (that is, by fixing the value of this indeterminate, we do not lose all of the JRS's volume in the cosine/sine dimensions, whereas the volume in the k^{v}-dimension goes to zero).

Let us say that $\overline{k_i^{\mathrm{v}}} = 0$ rad/s and $\Delta k_i^{\mathrm{v}} = \frac{\pi}{3}$ rad/s Therefore, $k_i^{\mathrm{v}} = \dot{\tilde{q}}_i = \frac{\pi}{6}$ corresponds to $(\dot{\tilde{q}}_i - \overline{k_i^{\mathrm{v}}})/\Delta k_i^{\mathrm{v}} = 0.5$. Then, $\texttt{slice}\left(J_i(t), \{\langle \kappa_i^{\mathrm{v}}(t) \rangle\}, \{0.5\}\right)$ returns the subset of $J_i(t)$ corresponding to setting $k_i^{\mathrm{v}} = \frac{\pi}{6}$ rad/s.

The forward occupancy map FO uses rotation matrices formed from the cosine and sine of each joint. By overapproximating these matrices, we can overapproximate FO. To this end, we represent sets of rotation matrices with matrix zonotopes.

We use each sliced zonotope $S_i(t)$ to produce a matrix zonotope $M_i(t)$ that overapproximates the rotation matrices for each joint i at each time t. We do so by forming 3D "rotation" matrices using the cosine and sine dimensions of the centers and generators of $S_i(t)$ (and keeping their indeterminates). We make use of the Rodrigues' rotation formula [195]. Let $\hat{u}_i = (u_{x,i}, u_{y,i}, u_{z,i})^\top \in \mathbb{R}^3$ be a unit vector denoting the axis of rotation for the i^{th} joint, and let U_i be the "cross-product matrix"

$$U_i = \begin{bmatrix} 0 & -u_{z,i} & u_{y,i} \\ u_{z,i} & 0 & -u_{x,i} \\ -u_{y,i} & u_{x,i} & 0 \end{bmatrix}, \tag{4.26}$$

then we write $M_i(t)$ as follows:

$$\begin{aligned} M_i(t) &= R_i(\tilde{q}_i) \left(X_i^{\mathrm{v}}(t), \left\{ G_i^{\mathrm{a}}(t), G_i^j(t) \right\}, \left\{ \langle \kappa_i^{\mathrm{a}}(t) \rangle, \langle \beta_i^j(t) \rangle \right\} \right)^{p(t)}, \\ X_i^{\mathrm{v}}(t) &= I_{3\times 3} + s_i^{\mathrm{v}} U_i + (1 - c_i^{\mathrm{v}}) U_i^2, \\ G_i^{\mathrm{a}}(t) &= s_i^{\mathrm{a}} U_i - c_i^{\mathrm{a}} U_i^2, \\ G_i^j(t) &= s_i^j U_i - c_i^j U_i^2. \end{aligned} \tag{4.27}$$

We call this the `makeMatZono` function in Alg. 2. Recall that each JRS is formed starting from an initial joint angle of 0 (the 2nd property of Def. 4.2.3). To account for the correct initial joint positions online, we simply rotate by $R_i(\tilde{q}_i)$ when forming the matrix zonotope $M_i(t)$.

To see how these formulae work in practice, we give the following example.

Example 4.4.8. *Consider the zonotope $S_i(t)$, and the rotation axis $\hat{u}_i = (0,0,1)^\top$ (a rotation about the vertical axis). Then, we have $U_i = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ and $U_i^2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. Plugging into the equations in* (4.27)*, we obtain*

$$X_i^{\mathrm{v}}(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + s_i^{\mathrm{v}} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + (1 - c_i^{\mathrm{v}}) \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \implies X_i^{\mathrm{v}}(t) = \begin{bmatrix} c_i^{\mathrm{v}} & -s_i^{\mathrm{v}} & 0 \\ s_i^{\mathrm{v}} & c_i^{\mathrm{v}} & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$G_i^{\mathrm{a}}(t) = s_i^{\mathrm{a}} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - c_i^{\mathrm{a}} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \implies G_i^{\mathrm{a}}(t) = \begin{bmatrix} c_i^{\mathrm{a}} & -s_i^{\mathrm{a}} & 0 \\ s_i^{\mathrm{a}} & c_i^{\mathrm{a}} & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$G_i^{j}(t) = s_i^{j} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - c_i^{j} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \implies G_i^{j}(t) = \begin{bmatrix} c_i^{j} & -s_i^{j} & 0 \\ s_i^{j} & c_i^{j} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Notice the similarity between the resulting centers and generators of $M_i(t)$ and a traditional rotation matrix about the vertical axis. Also a 1 appears in the $(3,3)$ position of $X_i^{\mathrm{v}}(t)$, while the generator matrices contain 0 in the $(3,3)$ position. This ensures that any element of $M_i(t)$ multiplied by a vector in $\mathbb{R}^3$ leaves the z-dimension unchanged, as desired.

Although the elements of $M_i(t)$ may not be orthonormal, the matrix zonotope is guaranteed to overapproximate the set of rotation matrices associated with the JRSs:

Lemma 4.4.9. *For any parameterized trajectory $\tilde{q} + q(\,\cdot\,; k) : T \to Q$ with $k^{\mathrm{v}} = \dot{\tilde{q}}$ and $k^{\mathrm{a}} \in K^{\mathrm{a}}$, every $R_i(\tilde{q}_i + q_i(t;k)) \in M_i(t)$.*

Proof. By [194, Thm. 3.3 and Prop. 3.7], all values attained by $\cos(q_i(t;k))$ and $\sin(q_i(t;k))$ are contained in each $J_i(t)$. By Alg. 1 and (4.24), each $S_i(t)$ overapproximates the values of sine and cosine of $q_i(t;k)$ for which $k_i^{\mathrm{v}} = \dot{\tilde{q}}_i$. For a given $q_i(t;k)$, $R_i(q_i(t;k))$ is given by plugging $\cos(q_i(t;k))$ and $\sin(q_i(t;k))$ into Rodrigues' formula. Therefore by applying Rodrigues' formula to $S_i(t)$ and rotating the output by $R_i(\tilde{q}_i)$, all possible $R_i(\tilde{q}_i + q_i(t;k))$ are contained in $M_i(t)$. ■

This lemma shows that for all parameterized configuration space trajectories starting at the initial position $\tilde{q}$ and velocity $\dot{\tilde{q}}$, ARMTD overapproximates the set of parameterized rotation matrices associated with these trajectories. Now we use $M_i(t)$ to overapproximate the link RS $\mathscr{L}_i$. Given the joint displacements l_i and link volumes L_i, we specify $l_j \in \mathbb{R}^3$ as a zonotope with center l_j and no generators, and L_i as a zonotope overapproximating the volume of link i. We multiply the matrix zonotopes $M_i(t)$ by L_i to overapproximate a swept volume, producing a *rotatotope* (Def. 4.4.3).

We use rotatotopes to overapproximate the forward occupancy map of each link by *stacking* rotatotopes representing link volume on top of rotatopes representing joint positions via a Minkowski sum:

Lemma 4.4.10. *For any $t \in T$ and $k \in K$, $\mathrm{FO}_i(\tilde{q} + q(t;k)) \subseteq V_i(t)$, where*

$$V_i(t) = \bigoplus_{j<i} \left(\prod_{n \leq j} M_n(t) \{l_j\} \right) \oplus \left(\prod_{n \leq i} M_n(t) L_i \right) \subset W. \tag{4.28}$$

Proof. First, note (4.28) is defined analogously to (4.2). We have $R_i(\tilde{q}_i + q_i(t;k)) \in M_i(t)$ from Lem. 4.4.9. The product of matrix zonotopes multiplied by a zonotope is a rotatotope by Lem. 4.4.5, and the Minkowski sum of rotatotopes are given exactly using Lem. 4.4.6. Therefore, all sets and operations in (4.28) are exact or conservative (note, we can overapproximate L_i with a zonotope), so $\mathrm{FO}_i(\tilde{q} + q(t;k)) \subseteq V_i(t)$. ■

Lem. 4.4.10 lets us overapproximate the RS: $\mathscr{L}_i \subseteq \bigcup_{t \in T} V_i(t) \implies \mathscr{L} \subseteq \bigcup_{t,i} V_i(t)$, as shown in Fig. 4.3. Alg. 2 computes $V_i(t)$.

Though $V_i(t) \subset W$, many of its generators are k^{a}-sliceable, because they are the product of k^{a}-sliceable matrix zonotope generators. Denote $V_i(t) = (\hat{x}_i(t), \hat{g}_i^j(t), \langle \hat{\beta}_i^j(t) \rangle)^{p(t)}$. Formally, the j^{th} generator $\hat{g}_i^j(t)$ is k^{a}-sliceable if there exists at least one $\langle \kappa_n^{\mathrm{a}}(t) \rangle \in \langle \hat{\beta}_i^j(t) \rangle$ with $n \leq i$. This means, by slicing by k^{a}, we can obtain a subset of $V_i(t)$ corresponding to that parameter. We make the distinction that a generator $\hat{g}_i^j(t)$ is *fully-k^{a}-sliceable* if *all* of its indeterminates are evaluated when sliced by k^{a}, i.e. $\langle \hat{\beta}_i^j(t) \rangle \subseteq \bigcup_{n \leq i} \langle \kappa_n^{\mathrm{a}}(t) \rangle$. Fully-$k^{\mathrm{a}}$-sliceable generators are created by multiplying k^{a}-sliceable generators with each other or with centers in (4.28). These generators are important because all of their indeterminates are evaluated by the trajectory optimization decision variable k^{a}, which we use in Sec. 4.4.3.2.

4.4.3 Constraint Generation

4.4.3.1 Theory

With the RS composed, we now use $\mathscr{L}$ to find all unsafe trajectory parameters $k \in K_{\mathrm{u}} \subseteq K$. These unsafe parameters violate joint limits, could cause collisions with obstacles, or could cause self-collisions. We treat K_{u} as a constraint for trajectory optimization, shown in Fig. 4.3. Recall

Algorithm 2 $\{V_i(t) : i = 1, \cdots, n_q,\ t \in T\} = \texttt{composeRS}(\tilde{q}, \dot{\tilde{q}})$

```
1:  parfor $t \in T$ // parallel for each time step
2:      for $i = 1 : n_q$ // for each joint
3:          $\kappa_i^{\mathrm{v}}(t) \leftarrow (\dot{\tilde{q}} - \overline{k_i^{\mathrm{v}}})/(\Delta k_i^{\mathrm{v}})$ // get value for (4.24)
4:          $S_i(t) \leftarrow \texttt{slice}(J_i(t), \langle \kappa_i^{\mathrm{v}}(t) \rangle, \kappa_i^{\mathrm{v}}(t))$ // slice JRS
5:          $M_i(t) \leftarrow \texttt{makeMatZono}(S_i(t), \tilde{q})$
6:          $V_i(t) \leftarrow M_i(t) L_i$ // init $V_i(t)$ for link volume RS
7:          $U_i(t) \leftarrow l_{i-1}$ // init rotatotope for joint location
8:          for $j = (i-1) : -1 : 1$ // predecessor joints
9:              $V_i(t) \leftarrow M_j^t V_i(t)$ // rotate link volume
10:             $U_i(t) \leftarrow M_j^t U_i(t)$ // rotate joint location
11:         end for
12:         for $j = (i-1) : -1 : 1$ // predecessor joints
13:             $V_i(t) \leftarrow V_i(t) \oplus U_j(t)$ // stack link on joints
14:         end for
15:     end for
16: end parfor
```

$q_{i,\mathrm{lim}}^-$, $q_{i,\mathrm{lim}}^+$, and $\dot{q}_{i,\mathrm{lim}}$ are joint limits. Let $\mathscr{O}$ be a set of obstacles. Further, we specify $I_{\mathrm{self}} \subset \mathbb{N}^2$ as a set of joint index pairs for which the links can intersect. That is, for $(i, j) \in I_{\mathrm{self}}$, there exist $q \in Q$ such that $\mathrm{FO}_i(q) \cap \mathrm{FO}_j(q) \neq \emptyset$. At each planning iteration, the unsafe trajectory parameters are $K_{\mathrm{u}} = K_{\mathrm{lim}} \cup K_{\mathrm{obs}} \cup K_{\mathrm{self}}$, where

$$K_{\mathrm{lim}} = \left\{k \mid \exists\, t \in T \text{ s.t. } q(t;k) < q_{i,\mathrm{lim}}^- \text{ or } q(t;k) > q_{i,\mathrm{lim}}^+ \text{ or } |\dot{q}(t;k)| > \dot{q}_{i,\mathrm{lim}}\right\} \tag{4.29}$$

$$K_{\mathrm{obs}} = \left\{k \mid Y \cap O \neq \emptyset,\ (Y, k) \in \mathscr{L},\ O \in \mathscr{O}\right\} \tag{4.30}$$

$$K_{\mathrm{self}} = \left\{k \mid Y_i \cap Y_j \neq \emptyset,\ (Y_i, k) \in \mathscr{L}_i,\ (Y_j, k) \in \mathscr{L}_j, (i, j) \in I_{\mathrm{self}}\right\}. \tag{4.31}$$

4.4.3.2 Implementation

We represent K_{lim} with functions $h_{\mathrm{lim}} : K^{\mathrm{a}} \to \mathbb{R}$. Notice in (4.5) that $q(t;k)$ is piecewise quadratic in k and $\dot{q}(t;k)$ is piecewise linear in k, so the parameterized trajectory extrema can be computed analytically. We construct h_{lim} from $\dot{\tilde{q}}_i$, $q_{i,\mathrm{lim}}$, and $\dot{q}_{i,\mathrm{lim}}$, such that $h_{\mathrm{lim}}(k^{\mathrm{a}}) < 0$ when feasible.

To represent K_{obs} (depicted in Fig. 4.3), first consider a particular k^{a}. We test if the corresponding subset of each rotatotope $V_i(t)$ could intersect any obstacle $O \in \mathscr{O}$. We overapproximate each O by a zonotope, which is always possible for compact, bounded sets [194] that appear in common obstacle representations such as octrees [196] or convex polytopes [197]. We formalize these conditions as constraints for online trajectory optimization. To proceed, we must test if two zonotopes intersect:

Lemma 4.4.11. *[198, Lem. 5.1] For two zonotopes $X = (x, g^i, \langle \beta^i \rangle)^n$ and $Y = (y, g^j, \langle \beta^j \rangle)^m$, $X \cap Y \neq \emptyset$ iff y is in the zonotope $X_{\mathrm{buf}} = (x, g^i, \langle \beta^i \rangle)^n \oplus (0, g^j, \langle \beta^j \rangle)^m$, where the subscript* *indicates X is buffered by the generators of Y.*

Since zonotopes are convex polytopes [198], by [194, Theorem 2.1], one can implement Lem. 4.4.11 by computing a *half-space representation* $(A_{\text{buf}}, b_{\text{buf}})$ of X_{buf} for which $A_{\text{buf}}z - b_{\text{buf}} \leq 0 \iff z \in X_{\text{buf}}$, where the inequality is taken elementwise. Using this representation, $X \bigcap Y = \emptyset \iff \max(A_{\text{buf}}y - b_{\text{buf}}) > 0$. We can use Lem. 4.4.11 for collision avoidance by replacing X (resp. Y) with a zonotope representing the arm (resp. an obstacle).

However, since we use rotatotopes, we need the following:

Lemma 4.4.12. *Any rotatotope MZ as in* (4.17) *can be overapproximated by a zonotope.*

So, we can overapproximate the intersection of each $V_i(t)$, sliced by k^{a}, with each $O \in \mathscr{O}$. Note, we only slice the fully-k^{a}-sliceable generators of $V_i(t)$, and treat all other generators conservatively by applying Lemma 4.4.12. That is, we do not slice any generators that have any indeterminates in addition to $\langle \kappa_i^{\text{a}}(t) \rangle$, and instead use those generators to (conservatively) buffer obstacles.

To check intersection, we separate $V_i(t)$ into two rotatotopes,

$$V_{i,\text{slc}}(t) = \left(x_i(t), g_{\text{slc}}^j, \langle \kappa_{\text{slc}}^j \rangle\right) \text{ and } V_{i,\text{buf}}(t) = \left(0, g_{\text{buf}}^n, \langle \beta_{\text{buf}}^n \rangle\right), \tag{4.32}$$

such that $V_i(t) = V_{i,\text{slc}}(t) \oplus V_{i,\text{buf}}(t)$, where $V_{i,\text{slc}}(t)$ has only fully-k^{a}-sliceable generators. That is, each $\langle \kappa_{\text{slc}}^j \rangle$ is a product of *only* $\langle \kappa_i^{\text{a}}(t) \rangle$ for one or more $i \in \{1, \cdots, n_q\}$. Note, the number of generators/indeterminates in $V_{i,\text{slc}}(t)$ and $V_{i,\text{buf}}(t)$ is omitted to ease notation. For any $k^{\text{a}} \in K^{\text{a}}$, since every generator of $V_{i,\text{slc}}(t)$ is k^{a}-sliceable, slicing $V_{i,\text{slc}}(t)$ by k^{a} returns a point. We express this with $\texttt{eval} : \mathcal{P}(W) \times K^{\text{a}} \to \mathbb{R}^3$ for which

$$\texttt{eval}(V_{i,\text{slc}}(t), k^{\text{a}}) = \texttt{slice}\left(V_{i,\text{slc}}(t), \left\{\langle \kappa_i^{\text{a}}(t) \rangle\right\}_{i=1}^{n_q}, \{\kappa(i)\}_{i=1}^{n_q}\right) \tag{4.33}$$

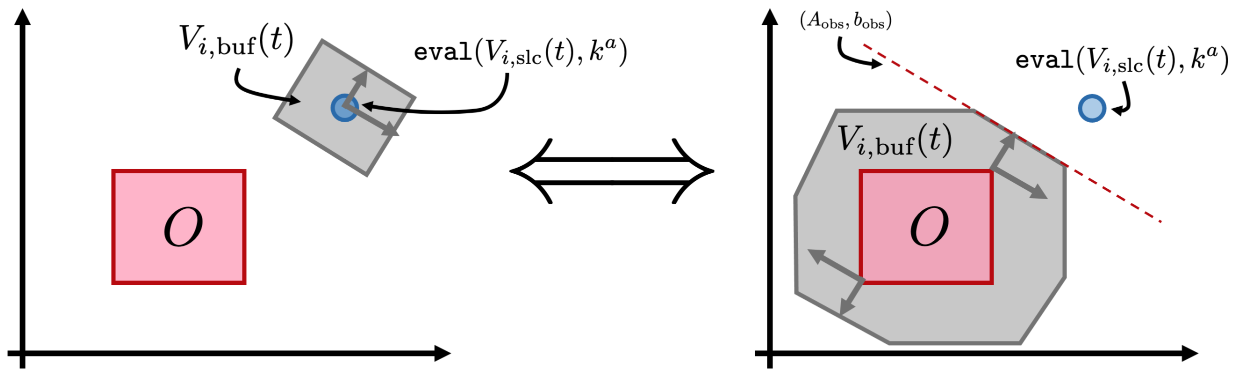
where $\kappa(i) = (k_i^{\text{a}} - \overline{k_i^{\text{a}}})/\Delta k_i^{\text{a}}$. Note, $\texttt{eval}$ can be implemented as the evaluation of polynomials.

Now, let A_{obs} and b_{obs} be the halfspace representation of $O_{\text{buf}} = O \oplus V_{i,\text{buf}}(t)$, and let $x = \texttt{eval}(V_{i,\text{slc}}(t), k^{\text{a}})$. Then,

$$(\{x\} \oplus V_{i,\text{buf}}(t)) \cap O = \emptyset \iff -\max\{A_{\text{obs}}x - b_{\text{obs}}\} < 0 \tag{4.34}$$

where $\{x\} \oplus V_{i,\text{buf}}(t)$ is overapproximated as a zonotope by applying Lem. 4.4.12. This process of generating the buffered obstacle O_{buf} and checking the intersection is illustrated in Fig. 4.4. We use (4.34) to overapproximate the parameters K_{obs} (4.30) with $h_{\text{obs}} : \mathbb{N} \times T \times \mathscr{O} \times K^{\text{a}} \to \mathbb{R}$ for which

$$h_{\text{obs}}(*, k^{\text{a}}) = -\max\left\{A_{\text{obs}}(*)\texttt{eval}(V_{i,\text{slc}}(t), k^{\text{a}}) - b_{\text{obs}}(*)\right\}. \tag{4.35}$$



$V_{i,\text{buf}}(t)$
$\texttt{eval}(V_{i,\text{slc}}(t), k^a)$
O
$(A_{\text{obs}}, b_{\text{obs}})$
$\texttt{eval}(V_{i,\text{slc}}(t), k^a)$
$V_{i,\text{buf}}(t)$
O

Figure 4.4: This figure depicts the obstacle constraint generation. On the left hand side, the obstacle O (pink zonotope) does not intersect the set $\{\texttt{eval}(V_{i,\text{slc}}(t), k^{\text{a}})\} \oplus V_{i,\text{buf}}(t)$ (blue dot $\oplus$ gray zonotope). In practice, we represent the constraint h_{obs} as depicted on the right hand side. The obstacle O is buffered by $V_{i,\text{buf}}(t)$, then turned into a halfspace representation A_{obs} and b_{obs}. If $-\max\{A_{\text{obs}}\texttt{eval}(V_{i,\text{slc}}(t), k^{\text{a}}) - b_{\text{obs}}\} < 0$ as shown, collision with the obstacle is impossible.

where $* = (i, t, O)$ for space. Here, $A_{\text{obs}}(i, t, O)$ and $b_{\text{obs}}(i, t, O)$ return the halfspace representation of $O \oplus V_{i,\text{buf}}(t)$. Importantly, for each obstacle, time, and joint, h_{obs} is a max of a linear combination of polynomials in k^{a} (per (4.33) and Alg. 1), so we can take its subgradient with respect to k^{a} [199] (also see [200, Thm. 5.4.5]). This constraint conservatively approximates K_{obs}:

Lemma 4.4.13. *If $k^{\text{a}} \in K_{\text{obs}}$, then there exists $i \in \mathbb{N}$, $t \in T$, and $O \in \mathscr{O}$ such that $h_{\text{obs}}(i, t, O, k^{\text{a}}) \geq 0$.*

Proof. This follows from Lem. 4.4.11 and Lem. 4.4.10; h_{obs} is positive when the zonotope produced by slicing $V_i(t)$ intersects O, and $V_i(t)$ provably contains all points in workspace reachable by the arm under the trajectory parameterized by k^{a}. ∎

We represent self-intersection constraints similarly, with a function $h_{\text{self}} : \mathbb{N} \times \mathbb{N} \times T \times K^{\text{a}} \to \mathbb{R}$. Suppose $(i, j) \in I_{\text{self}} \subset \mathbb{N}^2$ indexes a pair of links that could intersect, whose volume is overapproximated by $V_i(t)$ and $V_j(t)$. In analogy to (4.32), define

$$V_{\text{self}}(i, j, t) = V_{i,\text{slc}}(t) \oplus (-V_{j,\text{slc}}(t)) \quad \text{and} \tag{4.36}$$

$$V_{\text{buf}}(i, j, t) = V_{i,\text{buf}}(t) \oplus V_{j,\text{buf}}(t), \tag{4.37}$$

where $-V_{j,\text{slc}}(t)$ means the center and generators are multiplied by -1. Let $A_{\text{self}}(i, j, t)$ and $b_{\text{self}}(i, j, t)$ return the half-space representation of $V_{\text{buf}}(i, j, t)$. Then, using $*$ in place of the arguments (i, j, t) for space,

$$h_{\text{self}}(*, k^{\text{a}}) = -\max\left(A_{\text{self}}(*)\texttt{eval}(V_{\text{self}}(*), k^{\text{a}}) - b_{\text{self}}(*)\right). \tag{4.38}$$

As with h_{obs}, h_{self} is a max of a linear combination of polynomials in k^{a}, so we can take the subgradient with respect to k^{a}.

Lemma 4.4.14. *If $k^{\text{a}} \in K_{\text{self}}$, then there exists $(i,j) \in I_{\text{self}}$ and $t \in T$ such that $h_{\text{self}}(i,j,t,k^{\text{a}}) \geq 0$.*

4.4.4 Trajectory Optimization

4.4.4.1 Theory

ARMTD performs trajectory optimization over $K \setminus K_{\text{u}}$ for an arbitrary user-specified cost function $\phi : K \to \mathbb{R}$ (which encodes information such as completing a task). ARMTD attempts to solve the following within t_{plan}:

$$k_{\text{opt}} = \underset{k \in K}{\operatorname{argmin}} \{ \phi(k) \mid k \notin K_{\text{u}},\ k^{\text{v}} = \dot{\tilde{q}} \}. \tag{4.39}$$

If no solution is found in time, the robot tracks the fail-safe maneuever from its previous plan.

4.4.4.2 Implementation

We implement (4.39) as a nonlinear program, denoted `optTraj` in Alg. 3.

$$\begin{array}{lll} k_{\text{opt}} = \underset{k^{\text{a}} \in K^{\text{a}}}{\operatorname{argmin}} & \phi(k^{\text{a}}) & \\ \quad\text{s.t.} & h_{\text{obs}}(i,t,O,k^{\text{a}}) < 0 & \forall\, i \in \{1,\cdots,n_q\},\ t \in T,\ O \in \mathscr{O} \\ & h_{\text{self}}(i,j,t,k^{\text{a}}) < 0 & \forall\, (i,j) \in I_{\text{self}},\ t \in T \\ & h_{\text{lim}}(k^{\text{a}}) < 0 & \forall\, i \in \{1,\cdots,n_q\}. \end{array} \tag{4.40}$$

Theorem 4.4.15. *Any feasible solution to* (4.40) *parameterizes a trajectory that is collision-free and obeys joint limits over the time horizon T.*

Proof. This follows from the conservatism of h_{obs} in Lem. 4.4.13 and of h_{self} in Lem. 4.4.14, while h_{lim} can be represented exactly. ■

ARMTD uses Alg. 3 at each planning iteration. Recall that, without loss of generality, each iteration generates a plan over the time horizon $T = [0, t_{\text{f}}]$ (by shifting the current time to 0). Also recall, at each iteration, we allot t_{plan} s within which to find a new plan. The initial position and velocity of each joint in each iteration is the position and velocity, at time t_{plan}, of the trajectory plan of the previous iteration. ARMTD attempts to find a safe trajectory within t_{plan} by optimizing over a set of safe trajectory parameters; Thm. 4.4.15 ensures that any feasible solution is actually collision-free and obeys joint limits. If no safe trajectory is found within the allotted time, the arm executes the braking maneuver specified by the previous safe trajectory. Assuming the arm does

Algorithm 3 $q_{\text{plan}} = \texttt{makePlan}(\tilde{q}, \dot{\tilde{q}}, q_{\text{prev}}, \mathscr{O}, \phi)$

```
1: $\{V_i(t)\} \leftarrow \texttt{composeRS}(\tilde{q}, \dot{\tilde{q}})$ // Sec. 4.4.2
2: $(h_{\text{obs}}, h_{\text{self}}, h_{\text{lim}}) \leftarrow \texttt{makeCons}(\tilde{q}, \dot{\tilde{q}}, \mathscr{O}, \{V_i(t)\})$ // Sec. 4.4.3
3: // solve (4.40) within $t_{\text{plan}}$ or else return $q_{\text{prev}}$
4: $q_{\text{plan}} \leftarrow \texttt{optTraj}\left(\phi, h_{\text{obs}}, h_{\text{self}}, h_{\text{lim}}, t_{\text{plan}}, q_{\text{prev}}\right)$ // Sec. 4.4.4
```

not start in collision, this algorithm ensures that the arm is always safe (see [89, Remark 70] or [180, Theorem 1]).

4.5 Demonstrations

We now demonstrate ARMTD in simulation and on hardware using the Fetch mobile manipulator (Fig. 4.1). ARMTD is implemented in MATLAB, CUDA, and C++, on a 3.6 GHz computer with an Nvidia Quadro RTX 8000 GPU. See our video: youtu.be/ySnux2owlAA. Our code is available: github.com/ramvasudevan/arm_planning.

4.5.1 Implementation Details

4.5.1.1 Manipulator

The Fetch arm (Fig. 4.1) has 7 revolute DOFs [41]. We consider the first 6 DOFs, and treat the body as an obstacle. The 7th DOF controls end effector orientation, which does not affect the volume used for collision checking. We command the hardware via ROS [201] over WiFi.

4.5.1.2 Comparison

To assess the difficulty of our simulated environments, we ran CHOMP [2] via MoveIt [202] (default settings, straight-line initialization). We emphasize that CHOMP is not a receding-horizon planner [202]; it attempts to find a plan from start to goal with a single optimization program. However, CHOMP provides a useful baseline to measure the performance of ARMTD. To the best of our knowledge, no open-source, real-time receding-horizon planner is available for a direct comparison. Note, we report solve times to illustrate that ARMTD is real-time feasible, but the goal of ARMTD is not to solve as fast as possible; instead, we care about finding provably collision-free trajectories in the allotted time t_{plan}.

4.5.1.3 High-level Planner

Recall that ARMTD performs trajectory optimization using an arbitrary user-specified cost function. In this work, in each planning iteration, we create a cost function for ARMTD using an intermediate waypoint between the arm's current configuration and a global goal. These waypoints are generated by a high-level planner (HLP). Note, the RS and safety constraints generated by ARMTD are independent of the HLP, which is only used for the cost function. To illustrate that ARMTD can enforce safety, we use two different HLPs, neither of which is guaranteed to generate collision-free waypoints. First, a straight-line HLP that generates waypoints along a straight line between the arm and a global goal in configuration space. Second, an RRT* [203] that only ensures the arm's end effector is collision-free. Thus, **ARMTD can act as a safety layer on top of RRT***. Note, we allot a portion of t_{plan} to the HLP in each iteration, and give ARTMD the rest of t_{plan}. We cannot use CHOMP as a receding-horizon planner with these HLP waypoints, because it requires a collision-free goal configuration.

4.5.1.4 Algorithm Implementation

Alg. 2 runs at the start of each ARMTD planning iteration. We use a GPU with CUDA to execute Alg. 2 in parallel, taking approximately 10–20 ms to compose a full RS. The constraint generation step in Alg. 3 is also parallelized across obstacles and time steps (this takes approximately 10–20 ms for 20 obstacles).

We solve ARMTD's trajectory optimization (4.40) using IPOPT [204]. The cost function ϕ is $||q(t_{\text{f}}; k) - q_{\text{des}}||_2^2$, where q_{des} is the waypoint specified by the HLP (straight-line or RRT*) at each planning iteration. We compute analytic gradients/sub-gradients of the cost function and constraints, and evaluate the constraints in parallel. IPOPT takes 100–200 ms when it finds a feasible solution in a scene with 20 random obstacles.

4.5.1.5 Reduction of Generators

Creating the rotatotopes $V_i(t)$ in (4.28) requires multiplying generators together and storing their product. For example, a matrix zonotope described by 10 matrices (a center and 9 generators) multiplied by a zonotope described by 2 vectors (a center and 1 generator) yields a rotatotope described by 20 vectors (a center and 19 generators). Because this process is repeated for each joint, the number of generators theoretically required to represent each rotatotope grows exponentially with the number of joints. In practice, many of these generators are very small, and their effect can be overapproximated without adding much conservatism to the RS.

We conservatively approximate (4.28) by reducing the number of generators after each product, with a `reduce` function implemented as in [194, Proposition 2.2 and Heuristic 2.1]. The `reduce` function keeps the largest n_{red} generators according to a user-defined metric (we used the L^2-norm), then overapproximates the rest of the generators with an axis-aligned box. This ensures the number of rotatotope generators never exceeds a user-specified size. From Lem. 4.3.1, each $M_i(t)$ has k^{a}-sliceable generators. If a k^{a}-sliceable generator is chosen for reduction, we no longer consider it k^{a}-sliceable. This is a conservative approach, because slicing reduces the volume of a rotatotope in Alg. 1. A generator that is no longer k^{a}-sliceable cannot decrease the volume of the RS for any choice of k^{a}.

4.5.1.6 Hyperparameters

To reduce conservatism, we partition K_i^{v} into $n_{\text{JRS}} \in \mathbb{N}$ equally-sized intervals and compute one JRS for each interval. At runtime, for each joint, we pick the JRS containing the initial speed $\dot{q}_i$. In each JRS, we set $\Delta k_i^{\text{a}} = \max\left\{r_{a_2},\ r_{a_1}|\overline{k_i^{\text{v}}}|\right\}$, with $r_{a_1}, r_{a_2} > 0$ so the range of accelerations scales with the absolute value of the mean velocity of each JRS. This reduces conservativism at low speeds, improving maneuverability near obstacles.

We also use these values: $t_{\text{plan}} = 0.5$ s, $t_{\text{f}} = 1.0$ s, $\Delta t = 0.01$ s, $n_{\text{JRS}} = 400$, $\dot{q}_{i,\text{lim}} = \pi \frac{\text{rad}}{s}$, $\ddot{q}_{i,\text{lim}} = \pi/3 \frac{\text{rad}}{s^2}$, $\overline{k_i^{\text{a}}} = 0 \frac{\text{rad}}{s^2}$, $r_{a_1} = 1/3 s^{-1}$, and $r_{a_2} = \pi/24 \frac{\text{rad}}{s^2}$. For collision checking, we overapproximate the Fetch's links with cylinders of radius 0.146 m.

There are two hyperparameters that determine a tradeoff between conservatism and online planning speed (without impacting safety). The first is the density of the time partition for the JRS. That is, if we partition time more finely to generate the zonotope JRS, then it takes longer to generate and evaluate constraints at runtime (because we have to consider more zonotopes), but the JRS is also less conservative (so, the robot has more free space to move through).

The second hyperparameter is the range of parameters in the trajectory parameterization. A larger range produces a more conservative JRS, because the same number of zonotopes (determined by the time partition) must contain a larger range of joint angles achieved by all parameterized trajectories. We mitigate this problem in practice by precomputing many JRSs (in this work, we used 400), each of which has a narrow range of initial velocity parameters K_i^{v}. We choose the range of acceleration parameters K_i^{a} to vary with the velocity parameters, so that at higher speeds, there is a larger range of available control actions. This reduces conservatism at lower speeds so that ARMTD can maneuver tightly around obstacles.

Note, each JRS only takes around 1 s to compute, since it is only for a single joint, and for the low-dimensional cosine/sine dynamics. At runtime, to construct the RS of the entire arm, we first select the JRS (for each joint) containing the current initial velocity within its narrow range. Then, we slice by the exact initial velocity to produce the RS, and the corresponding collision-avoidance constraints.

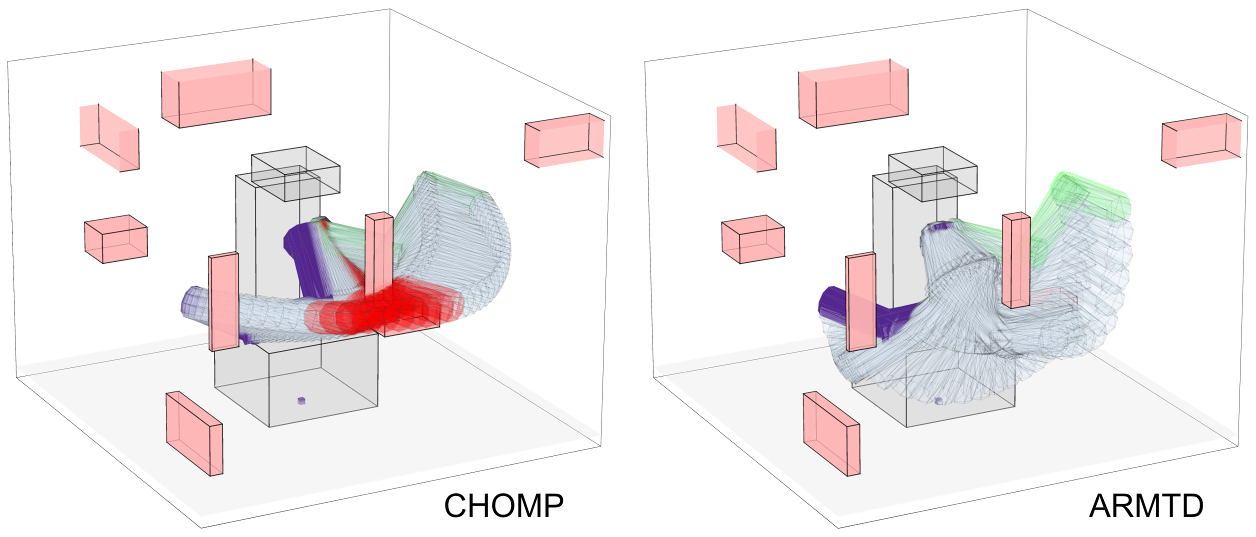


Figure 4.5: A Random Obstacles scene with 8 obstacles in which CHOMP [2] converged to a trajectory with a collision (collision configurations shown in red), whereas ARMTD successfully navigated to the goal (green); the start pose is shown in purple. CHOMP fails to move around a small obstacle close to the front of the Fetch.

4.5.2 Simulations

4.5.2.1 Setup

We created two sets of scenes. The first set, Random Obstacles, shows that ARMTD can handle arbitrary tasks (see Fig. 4.5). This set contains 100 tasks with random (but collision-free) start and goal configurations, and random box-shaped obstacles. Obstacle side lengths vary from 1 to 50 cm, with 10 scenes for each $n_O = 4, 8, ..., 40$.

The second set, Hard Scenarios, shows that ARMTD guarantees safety where CHOMP converges to an unsafe trajectory. There are seven tasks in the Hard Scenarios set: (1) from below to above a table, (2) from one side of a wall to another, (3) between two vertical posts, (4) from one set of shelves to another, (5) from inside to outside of a box on the ground, (6) from a sink to a cupboard, (7) through a small window. These scenarios are shown in Fig. 4.6.

4.5.2.2 Results

Table 4.1 presents ARMTD (with a straight-line HLP) and CHOMP's results for the Random Obstacles scenarios. ARMTD reached $84/100$ goals and had $0/100$ crashes, meaning ARMTD stopped safely $16/100$ times without finding a new safe trajectory. CHOMP reached $82/100$ goals and had $18/100$ crashes. CHOMP always finds a trajectory, but not necessarily a collision-free one; it can converge to infeasible solutions because it considers a non-convex problem with obstacles as areas of high cost (not as hard constraints). We did not attempt to tune CHOMP to only find feasible

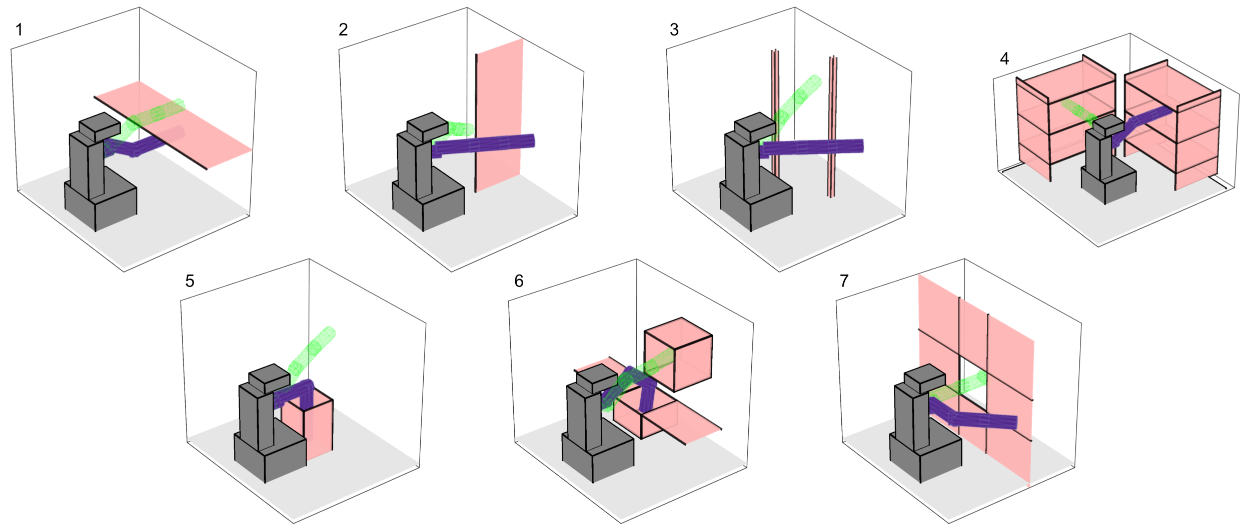


Figure 4.6: The set of seven Hard Scenarios (number in the top left), with start pose shown in purple and goal pose shown in green. There are seven tasks in the Hard Scenarios set: (1) from below to above a table, (2) from one side of a wall to another, (3) between two vertical posts, (4) from one set of shelves to another, (5) from inside to outside of a box on the ground, (6) from a sink to a cupboard, (7) through a small window.

plans (e.g., by buffering the arm), since this incurs a tradeoff between safety and performance. Note, in MoveIt, infeasible CHOMP plans are not executed (if detected by an external collision-checker).

We report the mean solve time (MST) of ARMTD over all planning iterations, while the MST for CHOMP is the mean over all 100 tasks. Directly comparing timing is not possible since ARMTD and CHOMP use different planning paradigms; we report MST to confirm ARMTD is capable of real-time planning (note that that ARMTD's MST is less than $t_{\text{plan}} = 0.5$).

We also report the mean normalized path distance (MNPD) of the plans produced by each planner (the mean is taken over all 100 tasks). The normalized path distance is a path's total distance (in configuration space), divided by the distance between the start and goal. For example, the straight line from start to goal has a (unitless) normalized path distance of 1. ARMTD's MNPD is 24% smaller than CHOMP's, which may be because CHOMP's cost rewards path smoothness, whereas ARMTD's cost rewards reaching an intermediate waypoint at each planning iteration (note, path smoothness could be included in ARMTD's cost function).

Table 4.2 presents results for the Hard Scenarios, displayed in Fig. 4.6. With the straight-line HLP, ARMTD does not complete any of the tasks but also has no collisions. With the RRT* HLP [203], ARMTD completes $5/7$ scenarios. CHOMP converges to trajectories with collisions in all of the Hard Scenarios.

Random Obstacles	% goals	% crashes	MST [s]	MNPD
ARMTD + SL	84	0	0.273	1.076
CHOMP	82	18	0.177	1.511

Table 4.1: MST is mean solve time (per planning iteration for ARMTD with a straight-line planner, total for CHOMP) and MNPD is mean normalized path distance. MNPD is only computed for trials where the task was successfully completed, i.e. the path was valid.

Hard Scenarios	1	2	3	4	5	6	7
ARMTD + SL	S	S	S	S	S	S	S
ARMTD + RRT*	O	O	O	S	O	S	O
CHOMP	C	C	C	C	C	C	C

Table 4.2: Results for the seven Hard Scenario simulations. ARMTD uses straight-line (SL) and RRT* HLPs. The entries are “O” for task completed, “C” for a crash, or “S” for stopping safely without reaching the goal.

4.5.3 Hardware

See our video: `youtu.be/ySnux2owlAA`. ARMTD completes arbitrary tasks while safely navigating the Fetch arm around obstacles in scenarios similar to Hard Scenarios (1) and (4). We demonstrate real-time planning by suddenly introducing obstacles (a box, a vase, and a quadrotor) in front of the moving arm. The obstacles are tracked using motion capture, and treated as static in each planning iteration. Since ARMTD performs receding-horizon planning, it can react to the sudden obstacle appearance and continue planning without crashing.

4.6 Conclusion

This chapter describes ARMTD; a real-time, receding-horizon manipulator trajectory planner with safety guarantees. The method composes novel parameterized reachable sets of arms which enable safety. ARMTD can enforce safety on top of an unsafe path planner such as RRT*, shown in both simulation and on hardware.

ARMTD has several limitations: it may not perform in real time without parallelization, is only demonstrated on 6-DOF planning problems, and has not yet been demonstrated planning around humans. However, because ARMTD uses time-varying reachable sets, it can readily extend to dynamic environments and planning with grasped objects. We can relax the assumption of static obstacles by treating obstacles as static only over each *time interval* of the JRS. Therefore, ARMTD shows great promise for planning safe trajectories around humans when accurate, conservative predictions of the volume occupied by a human in the workspace can be generated.

ARMTD guarantees the safety of the trajectories it produces; however, this requires an assumption (Assum. 4.2.4) that these trajectories can be perfectly executed by a robot. In practice, there may be some tracking error between the desired trajectory and the actual trajectory that the robot follows because of the robot's dynamics and actuator limits. We are currently extending ARMTD to guarantee safety in the presence of dynamic uncertainties by bounding the maximum tracking error possible for each joint, and buffering the JRSs to account for this error. This will be particularly useful for guaranteeing safety when grasping objects whose inertial properties are not exactly known.

Future work may extend ARMTD to more complicated robots such as walking robots, where safety is critical and difficult to achieve [169, 168]. Walking robots may prove effective at manipulation in the home because of their ability to navigate spaces built for humans [205].

The results in this work show promise for practical, safe robotic arm trajectory planning. In particular, ARMTD can be employed as a planner for manipulators operating as assistive robots, where guaranteees of collision-free operation are vital for operating around older persons and persons with disabilities.

CHAPTER 5

Safe Trip-recovery Trajectories for Robotic Prosthetic Legs to Prevent Falls

Portions of this chapter appear in the online preprint [96]:

Shannon M. Danforth, **Patrick D. Holmes**, and Ram Vasudevan. “Trip Recovery in Lower-Limb Prostheses using Reachable Sets of Predicted Human Motion”. *arXiv Preprint*, (2020).

5.1 Introduction

This chapter focuses on the task of **fall prevention** for wearable robot users. In 2005, an estimated $623,000$ people were living with major lower-limb loss in the United States, with the number predicted to double by 2050 [206]. A significant number of the lower extremity amputations performed each year occur in persons aged 65 or older [207]. The majority of these people use passive lower-limb prostheses, with instances of falling comparable to older adults over 85 [208, 209]. Powered lower-limb prostheses have the potential to lower metabolic cost and improve gait symmetry [56], and recent developments in their control allow them to assist users while walking up slopes, ascending/descending stairs, and walking backwards [58]. However, despite having the potential to assist users during slips or trips, most powered prostheses do not include stumble-recovery features. Equipping powered prostheses with stumble-recovery features could lead to fewer falls and increased balance confidence in lower-limb prosthesis users.

5.1.1 Related Work

Before discussing why stumble-recovery control is difficult in prostheses, it is useful to understand their general control framework. During locomotion, robotic lower-limb prostheses use a high-level controller to estimate the human’s intent. Common high-level controllers include activity mode detection [210, 211], direct volitional control [212], or a combination of both [213]. A mid-level controller then converts this estimated intent into a desired trajectory for a low-level controller to track [58].

Two challenges contribute to the difficulty of designing effective stumble-recovery controllers within this hierarchical control framework. The first is reliably detecting the stumble itself. In stumble experiments, humans have exhibited responses with latencies under 100 ms [214]. Researchers have shown that accelerometers mounted on the prosthesis can detect stumbles within this short time span, although false positives are common [215, 216]. The second challenge lies in choosing a recovery behavior for the prosthesis once the stumble has been detected. Difficulties in this area arise from stumbles leading to non-nominal joint trajectories, the presence of uncertainty in estimates of human intent, and the inability of existing mid-level controllers to generate trajectories for stumble recovery in real-time.

As a human undergoes a stumble and tries to recover, the joint trajectories necessary for a successful recovery may differ significantly from their nominal walking behavior [3]. Many controllers use an estimated human state to prescribe joint torques, such as virtual constraint control [217] and finite-state control [218]. Both methods express prosthetic target trajectories and torques as functions of gait phase instead of time, allowing for generalization across tasks, speeds, and users. However, following a stumble, the nominal trajectories utilized by these controllers based on an estimate of the person's phase may be inappropriate for recovery. This mismatch between the human's intent and the commanded trajectories of the robot could increase fall risk by producing unexpected torques [58].

Additionally, the variation in human stumble recoveries [3] makes it impossible to predict exactly how the human will respond. Recent work has proposed a fast quadratic program which shows great promise for planning post-stumble recoveries [219]. However, the method uses a single most likely estimate of human behavior instead of a set or distribution of predictions. Instead, a more robust approach would account for uncertainty by designing controllers that are suitable across a range of possible human behaviors. Generating trajectories online while accounting for a range of possible human stumble-recovery behavior is challenging, and requires optimizing over sets or distributions of trajectories.

5.1.2 TRIP-RTD Summary

With these challenges in mind, we present a method for planning a prosthetic recovery strategy online that accounts for a range of human behavior. In this chapter, we build on the ARMTD framework introduced in Chap. 4.

In the context of trip recovery, it is impossible to predict exactly how a person will respond to

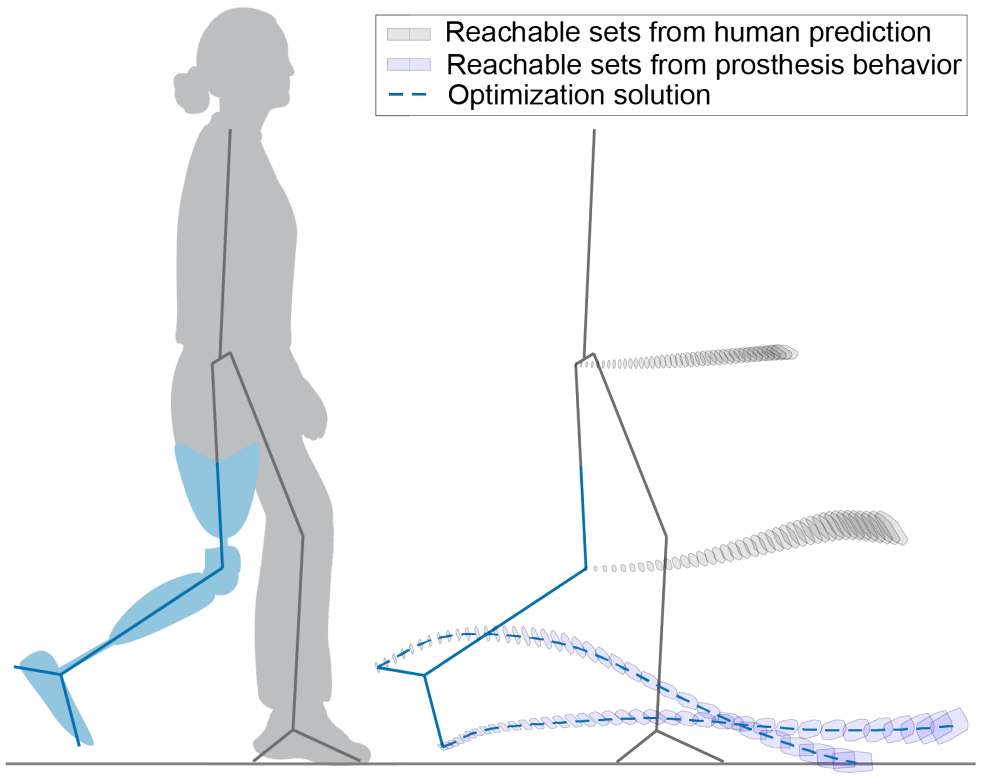

Reachable sets from human prediction
Reachable sets from prosthesis behavior
Optimization solution

Figure 5.1: An illustration of the TRIP-RTD framework, which plans trajectories for a prosthetic knee and ankle following a stumble while the prosthesis is in swing. At the time of the stumble, reachable sets corresponding to predicted human motion are generated (gray sets). The motion of the prosthetic heel and toe depend on both the motion of the person's hip and the motion of the prosthetic joints. TRIP-RTD composes reachable sets of heel and toe positions (shown in blue) from the predicted hip motion and parameterized prosthetic knee and ankle trajectories. These sets are used to develop safety constraints which avoid premature ground contact and ensure heel strike occurs at a time and place which guarantees successful recovery. TRIP-RTD solves an optimization problem subject to these constraints to generate a safe desired trajectory of the prosthetic leg.

a perturbation. Perhaps the best one can do is to generate a probabilistic prediction of possible trajectories the person may take. Then given this distribution of predicted trajectories, one should choose a plan for the prosthesis that *maximizes* the probability of the user's recovery. However, it is unclear how distributions over a person's hip angles, for example, propagate to possible positions of a prosthetic heel and toe in the workspace, which is necessary for avoiding unwanted collisions and placing the foot in an appropriate position. In this work, we therefore take a more restrictive view: given a set of predictions assumed to contain the actual human behavior, can we design a plan for the prosthesis which guarantees a safe recovery? We contend that while it would be terrible for a prosthesis to fail to prevent a fall, it would be worse for the prosthesis to cause a fall by trying to prevent one. Therefore, we focus on plans which are *incapable* of causing a fall, though finding such a plan may not always be feasible.

We apply a similar reachability-based approach to a model of a human with a transfemoral, unilateral amputation and a robotic knee and ankle prosthesis. We call our method **TRIP-RTD** (Trip Recovery in Prostheses via Reachability-based Trajectory Design). This work focuses on planning trajectories for a prosthesis in swing as the system undergoes a stumble caused by a trip. We assume that an accurate and conservative set-based prediction of the human's possible post-perturbation trajectories is available. As in ARMTD, we define a set of *trajectory parameters* which parameterize trajectories of the prosthetic joints. Then, we construct reachable sets of the human-prosthesis system based on the predicted human response to a perturbation and the prosthetic trajectory parameters. Online, we use these reachable sets with a *target set* of model states that ensure a successful recovery to generate constraints for an online optimization. We select prosthetic knee and ankle behavior that ensures the next heel strike happens at an appropriate time and location based on the predicted human motion. An overview of TRIP-RTD can be found in Fig. 5.1.

5.1.3 Contributions

We make the following contributions. First, in Sec. 5.4-5.5 we introduce a tractable framework for planning prosthetic stumble-recovery trajectories online that account for a range of predicted human behavior. Second, we demonstrate the method in simulation using human stumble-recovery data collected from able-bodied subjects and report results in Sec. 5.6. These experiments are described in Sec. 5.2. The remaining sections are Sec. 5.3 which describes preliminaries and Sec. 5.7 which discusses the method's effectiveness and plans for implementing the algorithm on hardware.

5.2 Tripping Dataset

We test TRIP-RTD on data collected during a previous experiment which studied trip recovery in people without amputations [3, 220].

In the preexisting dataset, subjects walked with tethers attached to their feet which were connected to a perturbation mechanism. The tethers were suddenly braked at various points in the swing phase, and motion capture data were collected to study the subject's responses. The authors in this work identified three distinct trip-recovery strategies, which they labeled as the "elevating", "delayed lowering", and "lowering" strategies [3]. These strategies are illustrated in Fig. 5.2. The subject's strategy selection depends largely on the point in the swing phase where they are tripped. Tripping earlier in swing often yielded an elevating strategy, where subjects would step up and over a perceived obstacle. Later in the swing phase the delayed lowering strategy becomes more prominent, where subjects would begin to lift their foot before placing it slightly behind the tripping position. Even later in the swing phase subjects often chose the lowering strategy, and moved their feet immediately backwards and down to the ground. Recently, researchers have confirmed these findings using data from a separate experiment and developed methods for accurately predicting a person's strategy using real-time measurable quantities [221].

We utilize this preexisting dataset to evaluate the TRIP-RTD framework in simulation. Using data from two non-amputee subjects participating in these experiments, we treat one of the subject's legs as a simulated "prosthesis". Our goal is to plan trajectories for this "prosthesis" using TRIP-RTD, and compare its performance to the actual observed responses.

In Sec. 5.3, we use kinematic data of an individual's trip-recovery trajectories to construct a library of parameterized trajectories as well as a target set representing successful recovery. This is done separately for each of the strategies introduced above. Let $s \in \{1, 2, 3\}$ index the elevating, delayed lowering, and lowering strategies, respectively. Then let $s^r, r \in \{1, 2, ..., n_r^s\}$ refer to the r-th observed trial for the s-th strategy, where a total of n_r^s trials for that strategy were observed.

5.3 Preliminaries

This section describes preliminaries necessary for the TRIP-RTD method. We discuss the planning model and a phase variable used to parameterize the swing phase. We construct a library of parameterized stumble-recovery joint angle trajectories from human data. Finally, we build a polytopic target set from data encapsulating states at heel strike that are observed to lead to nominal walking after a perturbation.

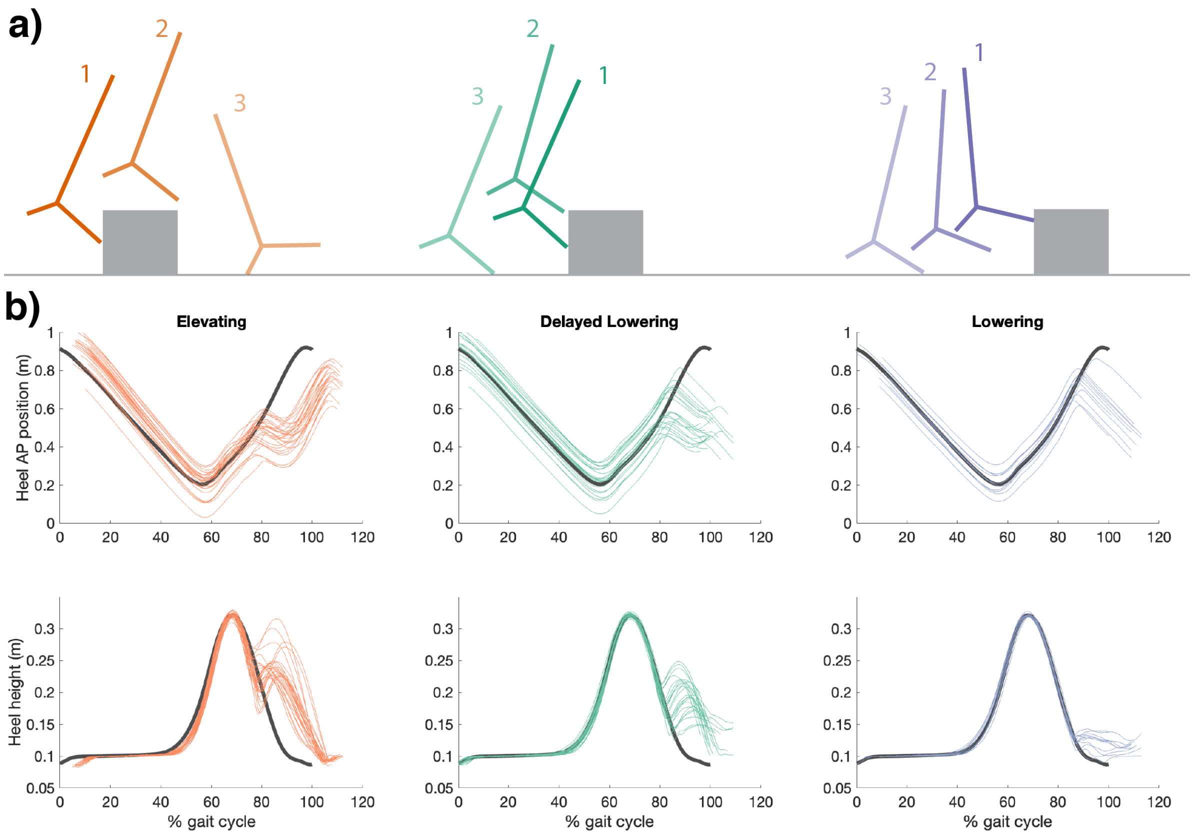


Figure 5.2: This figure illustrates the kinematic characteristics of the three distinct trip recovery strategies. In (a), stick figures demonstrate the temporal ordering of events. The elevating strategy (orange) involves upward and forward movement of the foot after a trip occurs. During the delayed lowering strategy (green), the foot initially moves upwards before being place behind the tripping position. Finally, with the lowering strategy (purple), the foot moves backwards and down. Data from a person without amputation being tripped at various points in their swing phase is shown in (b). These graphs illustrate observed trajectories of the three distinct trip recovery strategies in terms of heel positions as the gait cycle progresses [3].

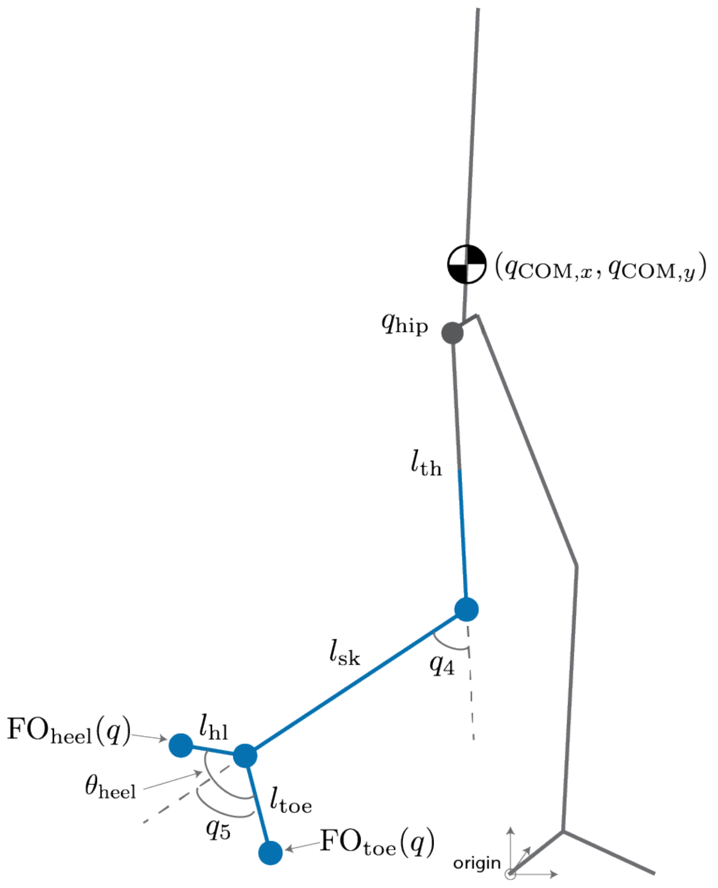


Figure 5.3: The kinematic planning model considered in this chapter of a transfemoral amputee (gray links) and robotic prosthetic leg (blue links). We assume the prosthetic leg is in swing when the trip occurs, and has a powered knee and ankle joint.

5.3.1 Planning Model

This chapter considers a 3D planning model of a robotic prosthetic leg used by a transfemoral amputee. We note that although a 2D version of this model is used to test the method in Sec. 5.6, we present the theory for this 3D model to keep the method general. We focus on trip recovery when the prosthetic leg is in swing. We consider a planning model representing single support with the intact limb in stance and the prosthetic limb in swing. A transition to double support happens instantaneously when the prosthetic heel or toe touches the ground.

The prosthetic leg is attached to the thigh of the body, and is composed of a powered knee and ankle joint. We model the prosthetic leg as a kinematic chain with two revolute joints, which rotate about the same (mediolateral) axis. This kinematic model is illustrated in Fig. 5.3. We treat the kinematic tree as composed of a *human subsystem* (the intact hip and thigh) and *prosthetic subsystem* (the prosthetic knee and lower leg).

We define an inertial reference frame with a coordinate system attached to the ground plane and located at the intact limb's stance heel position, which we assume to be constant in this dynamic mode. Let the coordinates $(x, y, z)^\top \in \mathbb{R}^3$ denote the anterior-posterior, mediolateral, and vertical components of positions in this coordinate frame, respectively. To be clear, the x-direction is forward, the y-direction is to the left, and the z-direction is up. We assume that the ground plane is flat and described by the normal vector $(0, 0, 1)^\top$, and since the coordinate system and the ground plane are coincident, $z = 0$ for points on the ground.

The movement of the prosthesis in the inertial reference frame is largely determined by the movement of the thigh to which the prosthesis is attached. We treat the thigh as a rigid link attached to the amputee's hip, which is described by six coordinates. We let $q_{\text{hip}} \in \mathbb{R}^3 \times \mathbb{S}^3$ be the configuration of the thigh, with

$$q_{\text{hip}} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{5.1}$$

where $(q_x, q_y, q_z)^\top \in \mathbb{R}^3$ describe the 3D position of the hip and $(q_1, q_2, q_3)^\top \in \mathbb{S}^3$ are Euler angles describing the flexion/extension, abduction/adduction, and internal/external rotation of the hip respectively. Next, let $q_p \in \mathbb{S}^2$ represent the two prosthetic joint angles:

$$q_p = \begin{bmatrix} q_4 \\ q_5 \end{bmatrix} \tag{5.2}$$

where q_4 is the knee angle and q_5 is the ankle angle. Finally, let $q = (q_{\text{hip}}, q_p)^\top \in \mathbb{R}^3 \times \mathbb{S}^5 = Q$ be the full configuration of the planning model with *configuration space* Q. Let Q_{hip} denote the configuration space of the hip configuration, and Q_p denote the configuration space of the prosthesis. Let W denote the *workspace*, i.e. the set of all points reachable by any point on the prosthetic leg in any configuration.

In this chapter, we are primarily concerned with position trajectories of the prosthetic heel and toe through the workspace. These positions depend on all the configuration variables defined above, as well as the lengths of the links of the kinematic tree. Let l_{th} denote the length of the thigh segment (i.e. the length from the intact hip to the prosthetic knee joint), and let l_{sk} denote the length of the prosthetic shank segment (i.e. the length from the knee joint to the ankle joint). Furthermore, we consider a prosthetic foot attached to the prosthetic ankle joint, and treat the heel and toe of this foot separately. We model the foot as composed of two rigid links, with one connecting the toe to the ankle and the other connecting the heel to the ankle. Let l_{hl} be the length from the ankle to the heel, and let l_{toe} be the length from the ankle to the toe. We define q_5 as the angle between the shank and the toe, and assume a constant angular offset θ_{heel} between the two links comprising the foot, as can be seen in Fig. 5.3. When we write these lengths in braces (i.e. $\{l_{\text{th}}\}$), we mean the point $\{(l_{\text{th}}, 0, 0)^\top\}$ (i.e. the vector pointing from the hip to thigh in a local coordinate frame, which has length l_{th}). By "stacking" the motion of the prosthetic leg on top of the motion of the hip, we obtain the positions of the prosthetic heel and toe. This process is exactly the same as the "forward occupancy" map FO defined for ARMTD (Sec. 4.2, (4.2)). To make this connection explicit, we define forward occupancy maps of the heel $\text{FO}_{\text{heel}} : Q \to \mathcal{P}(W)$ and toe $\text{FO}_{\text{toe}} : Q \to \mathcal{P}(W)$ as follows:

$$\text{FO}_{\text{heel}}(q) = \left\{ \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \right\} \oplus \prod_{i=1}^{3} R_i(q_i)\{l_{\text{th}}\} \oplus \prod_{i=1}^{4} R_i(q_i)\{l_{\text{sk}}\} \oplus \prod_{i=1}^{5} R_i(q_i)R(\theta_{\text{heel}})\{l_{\text{hl}}\} \tag{5.3}$$

$$\text{FO}_{\text{toe}}(q) = \left\{ \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \right\} \oplus \prod_{i=1}^{3} R_i(q_i)\{l_{\text{th}}\} \oplus \prod_{i=1}^{4} R_i(q_i)\{l_{\text{sk}}\} \oplus \prod_{i=1}^{5} R_i(q_i)\{l_{\text{toe}}\} \tag{5.4}$$

where as in (4.1), the notation $\prod_{i=1}^{3} A_i$ is understood as $A_1 A_2 A_3$, and $R_i(q_i) \in \text{SO}(3)$ is a rotation matrix corresponding to a rotation by q_i about the correct axis. Similarly, $R(\theta_{\text{heel}})$ is a constant rotation matrix representing the offset between the ankle angle and the heel link. Although the maps FO_{heel} and FO_{toe} yield a single point for a given configuration $q \in Q$, we have written them using set notation because later we overapproximate these maps using sets.

When identifying unsafe trajectories in Sec. 5.5, we rely on projections of these maps onto the ground plane and vertical axis. Let $\pi_{xy} : \mathbb{R}^3 \to \mathbb{R}^2$ project points onto the transverse plane (i.e. the x-y plane), and $\pi_z : \mathbb{R}^3 \to \mathbb{R}$ project points onto the vertical axis. Then, the position of the swing heel in the x-y plane is $\pi_{xy}(\text{FO}_{\text{heel}}(q))$, and the height of the swing heel is $\pi_z(\text{FO}_{\text{heel}}(q))$.

In Sec. 5.3.5, we use a person's center of mass (COM) state to describe a "target set" observed to lead to nominal walking. Let $q_{\text{COM}} = (q_{\text{COM},x}, q_{\text{COM},y})^\top \in \mathbb{R}^2$ denote a person's COM position in the transverse plane. Because the COM z-position remains mostly constant during walking [222], we do not consider its motion in this chapter.

5.3.2 Phase Variable and Nominal Trajectory

As described in the introduction, phase-based techniques have been successfully applied to control prostheses, and utilize a *phase variable* to convert human states into target joint trajectories. The phase variable must be a function of measurable quantities and monotonically non-decreasing.

Previous researchers have used the forward progression of the center of pressure during stance [223] and the thigh angle [213] as phase variables. Phase-based techniques specify a *nominal trajectory* which maps the phase variable to desired joint angles and velocities. In this work, we consider a nominal trajectory corresponding to level-ground walking. This nominal trajectory is tracked by the prosthesis during swing, i.e. using PD control [223]. We first formalize these concepts in theory then present our specific implementation.

5.3.2.1 Theory

Definition 5.3.1. *The phase variable t is a scalar-valued function of measurable quantities. The value of the phase variable is monotonically non-decreasing during the swing phase. Without loss of generality, let $t(\cdot) \in [0, 1]$, where 0 corresponds to the minimum observed phase and 1 corresponds to the maximum observed phase. For brevity, we write $t \in [0, 1]$, where the mapping from measurable quantities is implied.*

Throughout this chapter, we abuse notation and let $q : [0, 1] \to \mathcal{Q}$ denote a trajectory of the generalized coordinates that is parameterized by phase. For example, let $q(t)$ be the configuration at phase t. Let $q_i(t)$ be the value of the i^{th} coordinate at phase t, where $i \in \{x, y, z, 1, 2, 3, 4, 5, ...$ $...\text{COM},x, \text{COM},y\}$. Furthermore, let $\dot{q}(t)$ and $\dot{q}_i(t)$ refer to the derivatives of these coordinates *with respect to phase*.

We make the following assumptions about the nominal behavior of the prosthesis:

Assumption 5.3.2. *We assume that during unperturbed walking, a phase-based control scheme is used to control the prosthesis. In particular, when the prosthesis is in swing and unperturbed, we assume the prosthesis tracks a nominal trajectory, i.e. $q_p(t) = q_{p,\text{nom}}(t)$.*

Furthermore, we assume the following about trajectories of the COM and human subsystem:

Assumption 5.3.3. *Trajectories of the hip ($q_{\text{hip}}, \dot{q}_{\text{hip}}$) and COM ($q_{\text{COM}}, \dot{q}_{\text{COM}}$) are independent of trajectories of the prosthesis ($q_p, \dot{q}_p$) during the tripped swing phase. This requires that*

1. *The dynamics of q_{hip}, q_{COM} and their derivatives do not depend on q_p or its derivative.*

2. *The human does not change their behavior based on what the prosthetic leg does.*

The first part of the assumption is a reasonable approximation because the mass of the prosthetic leg is small compared to the mass of the amputee's body. The second part may be reasonable to assume in the context of trip recovery, where there is little time to sense and react to changes in prosthetic behavior.

5.3.2.2 Implementation

Presently, we use the percentage of swing phase as a phase variable. We say the swing phase starts at toe off ($t = 0$) and ends when the foot next makes contact with the ground ($t = 1$). To construct the nominal phase-based trajectory, we take the mean observed trajectories of the knee and ankle angles and their velocities (as a function of phase) during unperturbed swing phases. For perturbed swing phases, we perform an extra alignment step and shift the phase variable so that the peak foot height aligns with the nominal trajectory.

We briefly note that this particular scheme is not necessarily viable for a hardware implementation, since it assumes knowledge of when the foot next contacts the ground. However, the method presented in this chapter is indifferent to how the phase variable is constructed, and future work will explore the use of other real-time measurable phase variables. For example, the thigh angle or the angle of the COM with respect to the stance foot may prove useful for parameterizing swing phase.

5.3.3 Receding-Horizon Planning and Timing

During unperturbed walking, a powered prosthesis in swing following a nominal trajectory may be sufficient to prevent falls. However, when a person is suddenly perturbed, a different prosthesis response may be necessary to keep the user from falling. The goal of TRIP-RTD is to quickly design stumble-recovery trajectories in response to perturbation that guarantee a person will not fall (given a set of predicted hip and COM motion).

In Chap. 4, ARMTD planned in a receding-horizon fashion. ARMTD guaranteed safety by always including a fail-safe maneuver at the end of every parameterized trajectory, so that if the optimization problem (4.39) was infeasible, the previous plan's fail-safe maneuver would be executed. Unfortunately, in the context of stumble-recovery, this same fail-safe maneuver of "braking to a stop" is not necessarily safe, as it may place the foot in an undesirable position. Despite this, TRIP-RTD can still operate in a receding-horizon fashion by defaulting to the nominal prosthesis controller in case no feasible solution can be found. However, the nominal prosthesis controller may not be guaranteed to produce a safe recovery. We also note that each trajectory planned by TRIP-RTD should cover the duration from the current time until the next foot contact occurs.

We define the phase at which swing foot contact occurs:

Definition 5.3.4. *Swing foot contact occurs at phase $t_f \in [0, 1]$. Specifically, t_f is the first $t \in [0, 1]$ such that $\pi_z(\mathrm{FO}_{\mathrm{heel}}(q(t)) \leq 0$ or $\pi_z(\mathrm{FO}_{\mathrm{toe}}(q(t)) \leq 0$.*

Recall that $\pi_z(\mathrm{FO}_{\mathrm{heel}}(q(t))$ and $\pi_z(\mathrm{FO}_{\mathrm{toe}}(q(t))$ give the vertical position of the swing heel and toe at configuration $q(t)$, so t_f is the first time that the swing heel or toe breaks the ground plane.

During unperturbed walking, heel strike normally signals a transition to the next double support phase. However, for the delayed lowering and lowering trip recovery strategies, it is actually the swing toe that first makes contact with the ground. Therefore we more generally say that a transition to double support occurs when either the heel or toe strikes the ground.

TRIP-RTD only considers trajectories from the time a stumble is detected until t_f:

Assumption 5.3.5. *The trip begins at phase* $t_s \in [0,1]$*, is immediately detected, and the initial state of the prosthesis is known, i.e.* $q_p(t_s) = \tilde{q}_p$ *and* $\dot{q}_p(t_s) = \dot{\tilde{q}}_p$*. After the phase* $t_f \in [0,1]$ *when foot contact and a transition to double support occurs, the nominal prosthetic controller resumes control.*

After a stumble occurs, we allot a time within which TRIP-RTD must generate a safe plan. To ease exposition, we formulate this in terms of phase, where $t_{\text{plan}} \in \mathbb{R}$ is the allotted phase. Until t_{plan} has elapsed, the prosthesis follows the nominal trajectory. If TRIP-RTD finds a solution before $t_s + t_{\text{plan}}$, the prosthesis tracks this desired trajectory for $t \in [t_s, t_f]$. Otherwise, if no safe trajectory can be found in the current planning iteration, the prosthesis continues to follow the nominal trajectory. TRIP-RTD continually repeats this receding-horizon approach until heel strike occurs. For example, between $t_s + t_{\text{plan}}$ and $t_s + 2t_{\text{plan}}$ TRIP-RTD attempts to find a new solution using updated predictions of the human's hip and COM motion. If TRIP-RTD is initially unable to find a solution, this paradigm enables updated predictions in subsequent planning iterations to facilitate the discovery of a safe plan. It is possible that no safe plan can ever be discovered, which may happen if a perturbation is too large to conceivably recover from.

5.3.4 Parameterized Stumble-Recovery Trajectories

Following in the footsteps of Sec. 4.2.4, we introduce the notion of *trajectory parameters* that describe the evolution of the prosthesis' generalized coordinates over time. We introduce the theory, then our specific implementation.

5.3.4.1 Theory

We define trajectory parameters as follows:

Definition 5.3.6. *Let* $K \subset \mathbb{R}^{n_k}$ *be a compact set of* n_k trajectory parameters*, where each* $k \in K$ *maps to a desired trajectory of the generalized coordinates* $q_p : [0,1] \to Q_p$*. Let* $q_p(t;k)$ *be the configuration at phase* $t \in [0,1]$ *parameterized by* $k \in K$*. We require* $q_p(\cdot;k)$ *satisfy two properties. First,* $\dot{q}_p(\cdot;k)$ *is continuous and at least once-differentiable over times* $t \in [0,1]$*. WLOG, we require* $q_p(0;k) = 0$*.*

We let $q_p(0; k) = 0$ so that online we can construct reachable sets that use the initial configuration $\tilde{q}_p$ at the start of the stumble t_s, described in Sec. 5.5.

As an example parameterization, we may choose $k = (k_4, k_5) \in K \subset \mathbb{R}^2$ to parameterize a desired mapping from step height to trajectories of the knee and ankle joints. The goal of TRIP-RTD is to select a trajectory parameter $k \in K$ that yields trajectories of the prosthetic joints which are safe across a range of predictions of human motion.

As in ARMTD (Assum. 4.2.4), these parameterized trajectories are kinematic, not dynamic:

Assumption 5.3.7. *In this work, we assume that the prosthesis is capable of perfectly tracking the trajectories specified by the trajectory parameterization.*

Because the prosthesis is in swing, its motors only have to move its own mass when tracking these trajectories. In Sec. 5.5, we ensure commanded trajectories start with the correct initial post-perturbation positions. This makes tracking the trajectories easier, as there is no initial error between the desired and actual positions of the robotic leg.

5.3.4.2 Implementation

As a refresher, in Sec. 5.2 we describe three distinct trip recovery strategies. We let $s \in 1, 2, 3$ index the strategies, and let s^r refer to the r-th observed trial within that strategy. Let $q_4^{s^r}(t)$ and $q_5^{s^r}(t)$ be the observed knee and ankle joint angles at phase t observed for a specific trial. We fit a library of parameterized stumble recoveries to these observed trials. Specifically, we take the joint velocities observed in each trial, and fit polynomial functions of t and k which best match the observed data. We parameterize joint velocities so that we can create joint angle trajectories which start from $q_p(0; k) = 0$, as required in Def. 5.3.6, which will allow us to correctly incorporate the post-stumble configuration $\tilde{q}_p$ online.

First, we compute a *mean* velocity trajectory for each of the observed strategies for the knee and ankle joints. We refer to these as $\dot{q}^s_{4,\text{mean}}(t)$ and $\dot{q}^s_{5,\text{mean}}(t)$. Then, letting

$$\dot{q}^s_4(t; k) = \dot{q}^s_{4,\text{mean}}(t) + (a_0 + a_1 t + ... + a_5 t^5) + k_4(b_0 + b_1 t) \tag{5.5}$$

$$\dot{q}^s_5(t; k) = \dot{q}^s_{5,\text{mean}}(t) + (c_0 + c_1 t + ... + c_5 t^5) + k_5(d_0 + d_1 t) \tag{5.6}$$

we use linear regression to solve for the coefficients $a_0, ...a_5, b_0, b_1, c_0, ..., c_5, d_0, d_1$ that minimizes the difference from a strategy's observed trials in the least-squares sense. In other words, we find the coefficients which minimize the squared difference between $\dot{q}^s_4(t; k)$ and $\dot{q}^s_5(t; k)$ and each observed $\dot{q}_4^{s^r}(t)$ and $\dot{q}_5^{s^r}(t)$ for $r = 1, ...n^r_s$. To employ linear regression to fit these coefficients, we assign the values of k_4 and k_5 based on the observed trajectories. We assign both k_4 and k_5 values corresponding to the *step height* observed in a given trial.

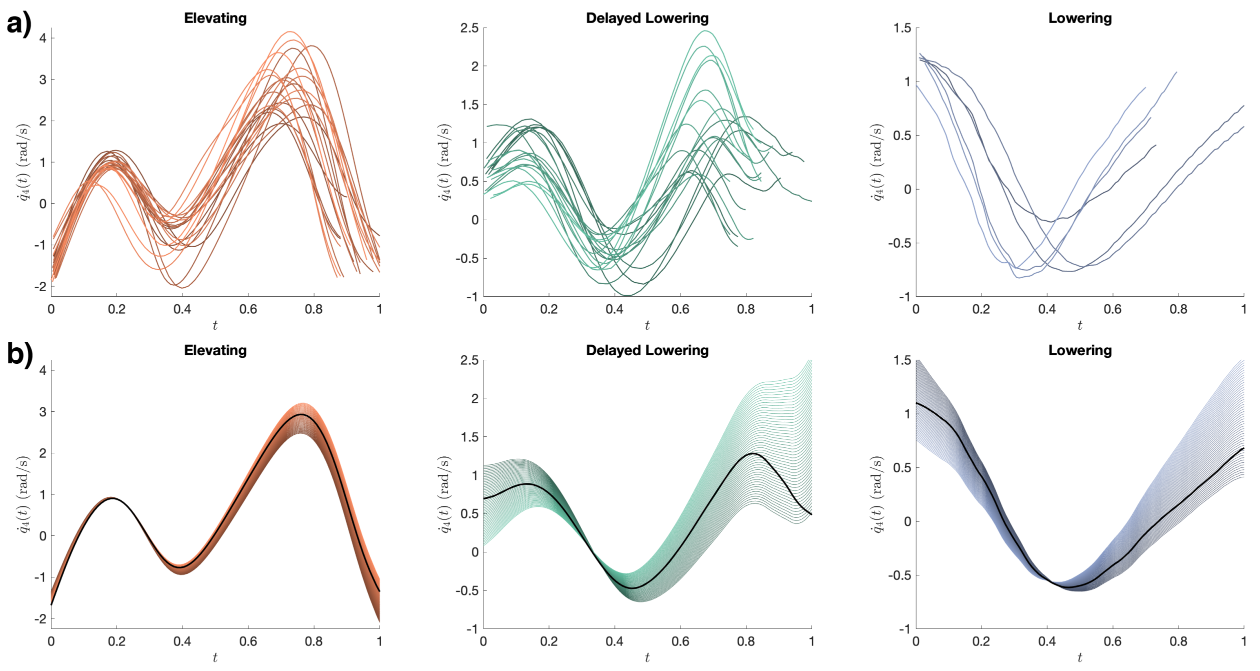


Figure 5.4: This figure shows example observed knee trajectories and parameterized trajectories for Subject ID 1. In (a), the observed knee velocity trajectories $\dot{q}_4^{s^r}(t)$ are plotted for each trip recovery strategy. The shade of each trajectory corresponds to the observed peak foot height in each trial, with darker shades corresponding to a lower peak foot height following a trip. In (b), the parameterization of knee velocity trajectories $\dot{q}_4(t;k)$ computed from the data in (a) is shown. The mean trajectory $\dot{q}^s_{4,\text{mean}}(t)$ is shown in black, and the gradient from dark to light is created by varying the value of k_4 from 0 to 1. A value of $k_4 = 0$ corresponds to a lower peak foot height, while a value of $k_4 = 1$ corresponds to a higher peak foot height. We note that in each of these plots, the phase $t \in [0, 1]$ has been scaled separately for each strategy.

WLOG, we normalize k_4 and k_5 so that $k_4 \in [0, 1]$, $k_5 \in [0, 1]$. To be clear, $k = (k_4, k_5)$ and therefore $K = [0, 1] \times [0, 1]$. The angles q_1, q_2, and q_3 correspond to the person's intact hip angles and are not parameterized by k.

These polynomials are fit separately to each s^{th} observed trip recovery strategy. An example knee trajectory paramterization is displayed in Fig. 5.4 and an example ankle knee trajectory paramterization is displayed in Fig. 5.5. Note that the trajectory parameterization in (5.5) and (5.6) is linear in $k \in K$. One may integrate these parameterized velocity trajectories from the initial condition $q_p(0; k) = 0$ to obtain position trajectories as specified in Def. 5.3.6.

5.3.5 Target Set

The goal of TRIP-RTD is to plan safe stumble-recovery trajectories. To characterize "safe recoveries" from a stumble, we create **target sets** which represent states at swing foot touchdown that

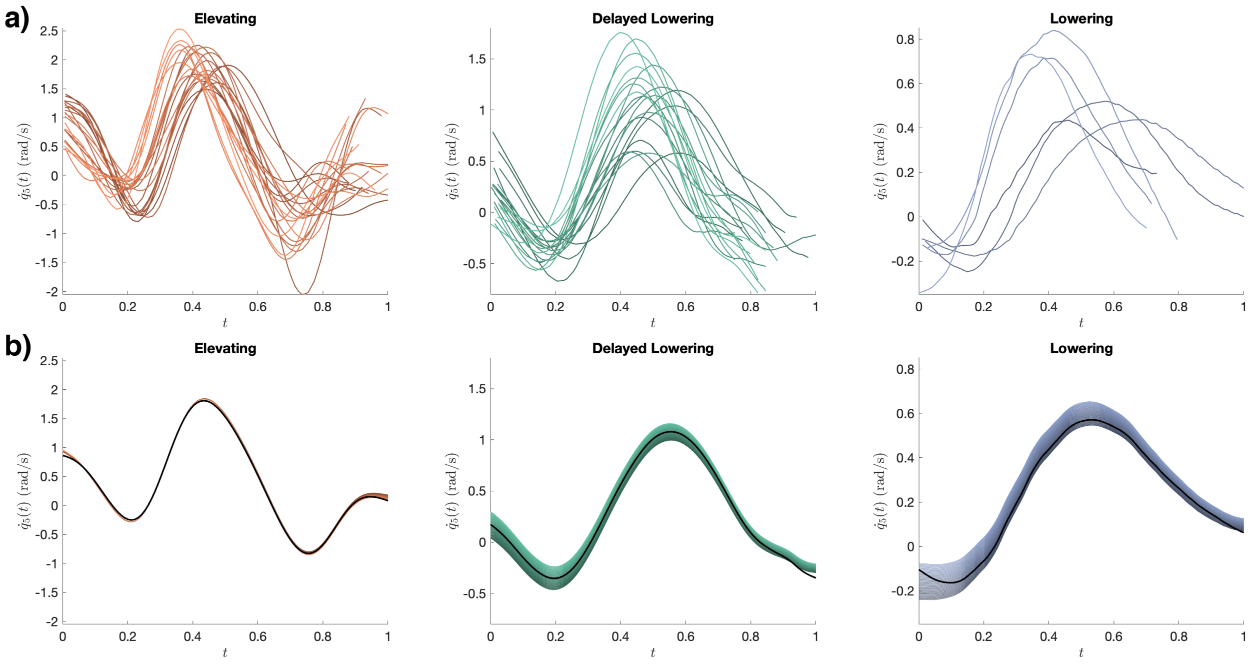


Figure 5.5: This figure shows example observed ankle trajectories and parameterized trajectories for Subject ID 1. In (a), the observed ankle velocity trajectories $\dot{q}_5^{s^r}(t)$ are plotted for each trip recovery strategy. The shade of each trajectory corresponds to the observed peak foot height in each trial, with darker shades corresponding to a lower peak foot height following a trip. In (b), the parameterization of ankle velocity trajectories $\dot{q}_5(t;k)$ computed from the data in (a) is shown. The mean trajectory $\dot{q}^s_{5,\text{mean}}(t)$ is shown in black, and the gradient from dark to light is created by varying the value of k_5 from 0 to 1. A value of $k_5 = 0$ corresponds to a lower peak foot height, while a value of $k_5 = 1$ corresponds to a higher peak foot height. We note that in each of these plots, the phase $t \in [0, 1]$ has been scaled separately for each strategy.

are known to be safe.

5.3.5.1 Theory

Definition 5.3.8. *A* target set $X_T \subset \mathbb{R}^6$ *is a set of COM positions* q_{COM}*, velocities* $\dot{q}_{\text{COM}}$*, and swing heel/toe positions on the ground plane at the time of foot contact that lead to steady-state walking when using the nominal prosthetic leg controller. The target sets may differ for each trip recovery strategy. Its formulation in terms of heel or toe positions depends on which portion of the foot first makes contact with the ground under that strategy. Recall that in our formulation, the foot contact phase is* t_f *and the* x*-*y *swing heel/toe position for a given configuration* $q(t)$ *is* $\pi_{xy}(\text{FO}_{\text{heel}}(q(t)))$

or $\pi_{xy}(\mathrm{FO}_{\mathrm{toe}}(q(t)))$. *Then, if*

$$\begin{bmatrix} q_{\mathrm{COM}}(t_f) \\ \dot{q}_{\mathrm{COM}}(t_f) \\ \pi_{xy}(\mathrm{FO}_{\mathrm{heel}}(q(t_f))) \end{bmatrix} \in X_T \quad \text{or} \quad \begin{bmatrix} q_{\mathrm{COM}}(t_f) \\ \dot{q}_{\mathrm{COM}}(t_f) \\ \pi_{xy}(\mathrm{FO}_{\mathrm{toe}}(q(t_f))) \end{bmatrix} \in X_T \tag{5.7}$$

the nominal prosthesis controller brings the amputee to steady-state walking.

Since the z-position of the swing heel/toe determines the foot contact phase t_f, we need only consider x-y positions of the swing heel and toe when defining X_T.

5.3.5.2 Implementation

Similar to the construction of target sets in Sec. 3.6, we build target sets from data observed to lead to successful walking following a perturbation. Recent work has shown that the position of the swing foot at heel strike following perturbation is predicted well by a linear map of COM positions and velocities [224]. Therefore, a polytope over COM states and foot contact positions is a natural choice for describing the target set. We fit separate target sets for each strategy, because the states at foot contact may differ significantly by strategy. For example, the elevating and lowering strategies may place the foot at very different positions. Target sets computed for Subject ID 1 are displayed in Fig. 5.6.

In practice, we treat the anterior-posterior and mediolateral states separately, and create two polytopes whose union forms the target set. Specifically, we form $X_{T,\mathrm{AP}}$ and $X_{T,\mathrm{ML}}$ using zonotopes with 3 generators each, which encompass all states at foot contact observed to lead to successful walking. This mirrors the method found in Sec. 3.6 for generating zonotopes from sets of points [164].

5.4 Offline Reachability Analysis

As in ARMTD (Chap. 4), TRIP-RTD uses parameterized trajectories of joint angles for planning. Rather than repeat the material presented in 4.3, we reference Chap. 4 extensively and adapt notation when necessary. Again, we compute **joint reachable sets** (JRSs) of parameterized prosthetic joint angle trajectories. These JRSs will be used online with reachable sets of predicted human motion in Sec. 5.5. All computations in this section are performed offline.

To be clear, in this work we use the term "reachable sets" to refer to sets of *kinematic* trajectories. We use these reachable sets to produce desired trajectories that the prosthesis attempts to track via a tracking controller, as in Assum. 5.3.7.

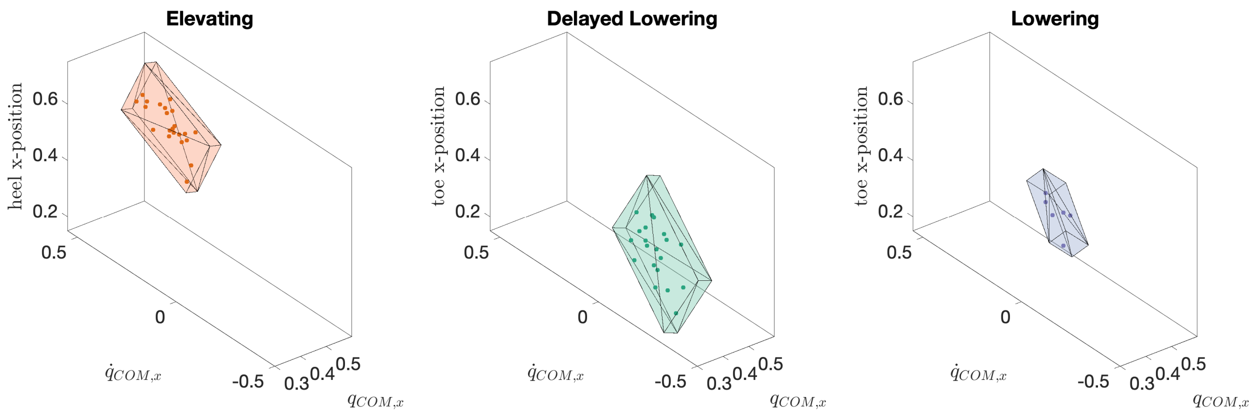


Figure 5.6: This figure shows the anterior-posterior target sets $X_{T,\text{AP}}$ computed from data for Subject ID 1. Note that the target sets are defined over heel positions for the elevating strategy and over toe positions for the delayed lowering and lowering strategies. The target sets found for each strategy occupy distinct regions. As expected for the elevating strategy, $X_{T,\text{AP}}$ is characterized by more forward progression of the heel and COM positions, and a higher COM velocity. For the delayed lowering strategy, $X_{T,\text{AP}}$ occupies a region of low forward progression of the toe and COM positions, and negative COM velocities. The lowering strategy's target set lies between these two.

5.4.1 Theory

Recall that the prosthetic knee and ankle angles are described by q_4 and q_5. We examine the evolution of $\cos(q_i)$ and $\sin(q_i)$ where $i \in \{4, 5\}$. By Def. 5.3.6, $q_p(\cdot; k)$ is at least once-differentiable, and therefore we can write the dynamics of $\cos(q_i)$ and $\sin(q_i)$ as a differential equation, where k is a constant, as in (4.8).

Then as in (4.9), we define the joint reachable sets:

$$\begin{aligned}\mathscr{J}_i = \Big\{&(c, s, k, t) \in \mathbb{R}^2 \times K \times [0, 1] \mid \exists\, t \in [0, 1] \text{ s.t. } q_i \text{ as in Def. 5.3.6,} \\ &c = \cos(q_i(t; k)),\ s = \sin(q_i(t; k)), \\ &\text{and } \tfrac{d}{dt}\big(\cos(q_i(t; k)), \sin(q_i(t; k)), k\big) \text{ as in (4.8)}\Big\}\end{aligned} \tag{5.8}$$

where $i = 4, 5$.

5.4.2 Implementation

We adopt the notation from Sec. 4.3.1.2, and represent the JRSs using zonotopes $Z = (x, g^i, \langle\beta^i\rangle)^p$. Again, we represent $\mathscr{J}_i$ using one zonotope per time interval. To compute $\mathscr{J}_i$, we make an initial condition zonotope $J_i(0) \subset \mathbb{R}^2 \times K$.

$$J_i(0) = \left(\tilde{x}, \left\{ \tilde{g}^k \right\}, \left\{ \langle \kappa_i \rangle \right\} \right), \tag{5.9}$$

with $\tilde{x} = [1, 0, 0.5]^\top$ and $\tilde{g}^k = [0, 0, 0.5]^\top$. The indeterminate $\langle \kappa_i \rangle$ corresponds to $\tilde{g}^k$, which represents $k_i \in [0, 1]$. As in Sec. 4.3.1.2, we use CORA [102] with the initial condition set $J_i(0)$ (5.9) as well as the parameterized joint velocities given in (5.5) and (5.6). The output of CORA is a sequence of zonotopes $J_i(t)$ which are guaranteed to overapproximate (5.8).

As in Lem. 4.3.1, each $J_i(t)$ has only one generator with non-zero element in the k_i dimension. For each $J_i(t)$ we denote the center $x_i(t)$, the generators $\{g_i^k(t), g_i^j(t)\}$ and the corresponding indeterminates $\{\langle \kappa_i \rangle, \langle \beta_i^j(t) \rangle\}$ for $j = 1, ..., p(t) \in \mathbb{N}$. As in Sec. 4.3.1.2 we write $p(t)$ since the number of generators is not necessarily the same for each $J_i(t)$. The generator $g_i^k(t)$ is *sliceable* (see Sec. 4.4.1) and lets us obtain subsets of the JRSs corresponding to particular choices of the parameters k.

5.5 Online Planning

5.5.1 Predicted Reachable Sets of the Human Subsystem

Because it is impossible to perfectly predict human behavior, we want to choose a desired trajectory of the prosthesis (represented by $k \in K$) that is robust to a range of plausible human stumble-recovery responses. In particular, Sec. 5.5.4 describes our method for choosing $k \in K$ that avoids premature swing foot contact with the ground and places the foot within a desired set.

First, we introduce the theory for representing sets of predictions of the human hip and COM following a stumble. We begin with the hip, which affects the motion of the prosthetic heel and toe as described in (5.3) and (5.4). The hip configuration q_{hip} (defined in (5.1)) is comprised of both translational and rotational components. The translational components shift the base of our kinematic tree, while we treat the rotational components similarly to the JRSs (but without a parameterization by k). First, we define the reachable set of plausible hip translations:

$$\begin{split} \mathscr{J}_{\text{xyz}} = \Big\{ (x, y, z, t)^\top \in \mathbb{R}^3 \times [0, 1] \mid t \in [t_s, 1], \\ x = q_x(t),\ y = q_y(t),\ z = q_z(t) \Big\}. \end{split} \tag{5.10}$$

Next, we define reachable sets of cosines and sines of the hip Euler angles. Recall that $q_1(t)$, $q_2(t)$, and $q_3(t)$ represent the flexion/extension, abduction/adduction, and internal/external rotation of the hip at phase t, respectively.

Then, we let

$$\begin{aligned}\mathscr{J}_i = \Big\{&(c, s, t)^\top \in \mathbb{R}^2 \times [0,1], \mid t \in [t_s, 1], \\ &c = \cos(q_i(t)),\ s = \sin(q_i(t))\Big\}\end{aligned} \tag{5.11}$$

for $i = 1, 2, 3$. Note the similarity to (5.8), but without the appearance of any trajectory parameters.

Finally, we use predictions of COM motion to help define safe foot placement. We define reachable sets of COM positions and velocities as follows:

$$\begin{aligned}\mathscr{J}_{\text{COM}} = \Big\{&(x, dx, y, dy, t)^\top \in \mathbb{R}^4 \times [0,1] \mid t \in [t_s, 1], \\ &x = q_{\text{COM},x}(t),\ dx = \dot{q}_{\text{COM},x}(t),\ y = q_{\text{COM},y}(t),\ dy = \dot{q}_{\text{COM},y}(t)\Big\}.\end{aligned} \tag{5.12}$$

5.5.1.1 Implementation

We overapproximate $\mathscr{J}_{\text{xyz}}$, $\mathscr{J}_{\text{COM}}$, and $\mathscr{J}_i$ for $i = 1, 2, 3$ using zonotopes. Specifically, let $\mathscr{J}_{\text{xyz}} \subseteq \bigcup_{t \in [t_s, 1]} J_{\text{xyz}}(t)$, $\mathscr{J}_{\text{COM}} \subseteq \bigcup_{t \in [t_s, 1]} J_{\text{COM}}(t)$ and $\mathscr{J}_i \subseteq \bigcup_{t \in [t_s, 1]} J_i(t)$ for $i = 1, 2, 3$, where each $J_{(\cdot)}(t)$ is a zonotope of the appropriate size.

Though this chapter makes use of these predictions, generating the predictions in the first place is an unsolved challenge. Researchers have made steps towards predicting some of these trajectories in real time [219]. However, for this chapter, we make the assumption that these sets are available to us, and save this challenge for future work.

Assumption 5.5.1. *At the time of the stumble, zonotopes $J_{\text{xyz}}(t)$, $J_{\text{COM}}(t)$ and $J_i(t)$ for $i = 1, 2, 3$ are immediately known for $t \in [t_s, 1]$ and overapproximate the true reachable sets $\mathscr{J}_{\text{xyz}}$, $\mathscr{J}_{\text{COM}}$, and $\mathscr{J}_i$ for $i = 1, 2, 3$.*

When working with the preexisting dataset described in Sec. 5.2, we form these zonotopes from observed hip and COM motion.

5.5.2 Reachable Set Construction

5.5.2.1 Theory

Equipped with predictions for the hip and COM motion, and with the precomputed JRSs of the prosthesis' joints, we now describe our method for combining these to compose reachable sets of the prosthetic heel and toe through workspace online. The following subsections will rely on these reachable sets to enforce ground clearance and desired foot placement constraints. These sets are a function of both the hip predictions as well as the trajectory parameters $k \in K$.

We denote the heel reachable set as $\mathscr{L}_{\text{heel}}$ and the toe reachable set as $\mathscr{L}_{\text{toe}}$, where

$$\begin{aligned}
\mathscr{L}_{\text{heel}} = \Bigg\{ (X, k, t) \subset \mathcal{P}(W) \times K \times [0,1] \,\Bigg|\, & t \in [t_s, 1], \\
& X = \text{FO}_{\text{heel}}(q(t;k)), \\
& q(t;k) = (q_x(t), q_y(t), q_z(t), q_1(t), q_2(t), q_3(t), q_4(t;k), q_5(t;k))^\top, \\
& (q_x(t), q_y(t), q_z(t), t) \in \mathscr{J}_{\text{xyz}}, \\
& (\cos(q_i(t)), \sin(q_i(t), t) \in \mathscr{J}_i \; \forall i \in \{1,2,3\} \\
& (\cos(q_i(t;k)), \sin(q_i(t;k), k, t) \in \mathscr{J}_i \; \forall i \in \{4,5\} \Bigg\}
\end{aligned} \tag{5.13}$$

where $\mathscr{L}_{\text{toe}}$ is similarly defined but using $\text{FO}_{\text{toe}}(q(t;k))$.

5.5.2.2 Implementation

It is important that we overapproximate $\mathscr{L}_{\text{heel}}$ and $\mathscr{L}_{\text{toe}}$ to guarantee safety when planning. We have formed zonotopes which overapproximate $\mathscr{J}_{\text{xyz}}$ and $\mathscr{J}_i$ for $i = 1, 2, 3, 4, 5$, but now we use these sets to overapproximate the forward occupancy maps FO_{heel} and FO_{toe}. Similar to Sec. 4.4.2.2, we overapproximate sets of rotation matrices from the cosine and sine of each rotational joint. In contrast to Chap. 4, however, we also have to account for the translation of the reachable sets due to the motion of the base of the kinematic chain (the hip position).

Almost identically to (4.27), we form matrix zonotopes from each $J_i(t)$. As in Sec. 4.4.2.2, let $\hat{u}_i$ denote the unit vector which the i-th joint rotates about, and U_i be its cross-product matrix (see (4.26)). For $i = 1, 2, 3$ (i.e. the hip angles), each $J_i(t) \subset \mathbb{R}^2$ describes sets of possible sines and cosines of the hip joint angles. We write $J_i(t) = (x_i(t), \{g_i^j(t)\}, \{\langle \beta_i^j(t) \rangle\})^{p(t)}$, where $x_i(t) = [c_i^x, s_i^x]^\top$ and $g_i^j(t) = [c_i^j, s_i^j]^\top$ and $p(t) \in \mathbb{R}$ is the number of generators. Then, we create matrix zonotopes $M_i(t)$

$$\begin{aligned}
M_i(t) &= \left(X_i(t), \left\{G_i^j(t)\right\}, \left\{\langle \beta_i^j(t) \rangle\right\}\right)^{p(t)}, \\
X_i(t) &= I_{3\times 3} + s_i^x U_i + (1 - c_i^x) U_i^2, \\
G_i^j(t) &= s_i^j U_i - c_i^j U_i^2
\end{aligned} \tag{5.14}$$

for $i = 1, 2, 3$. We call this the `makeHipMatZono` function in Alg. 4. For an explainer of these formulae, please refer to Example 4.4.8 in Sec. 4.4.2.2.

Algorithm 4 $\{V_{\text{heel}}(t), V_{\text{toe}}(t) \,:\, t \in T\} = \texttt{composeRS}(\tilde{q}_p, q_{p,\text{nom}})$

```
 1: for $t \in T$ // parallel for each time step
 2:     for $i = 1:3$ // for each hip angle
 3:         $M_i(t) \leftarrow \texttt{makeHipMatZono}(J_i(t))$ // see (5.14)
 4:     end for
 5:     for $i = 4:5$ // for each prosthetic joint angle
 6:         $M_i(t) \leftarrow \texttt{makeProsMatZono}(J_i(t), \tilde{q}_p, q_{p,\text{nom}})$ // see (5.16)
 7:     end for
 8:         $V_{\text{heel}}(t) \leftarrow \prod_{i=1}^{3} M_i(t) l_{\text{th}} \oplus \prod_{i=1}^{4} M_i(t) l_{\text{sk}} \oplus \prod_{i=1}^{5} M_i(t) R(\theta_{\text{heel}}) l_{\text{hl}}$
 9:         $V_{\text{toe}}(t) \leftarrow \prod_{i=1}^{3} M_i(t) l_{\text{th}} \oplus \prod_{i=1}^{4} M_i(t) l_{\text{sk}} \oplus \prod_{i=1}^{5} M_i(t) l_{\text{toe}}$
10:     end for
11:     $V_{\text{heel}}(t) \leftarrow V_{\text{heel}}(t) \oplus J_{\text{xyz}}(t)$ // add hip motion
12:     $V_{\text{toe}}(t) \leftarrow V_{\text{toe}}(t) \oplus J_{\text{xyz}}(t)$ // add hip motion
13: end for
```

For the prosthetic knee and ankle joints, this process is only slightly different: we account for the fact that the JRSs may not start from the correct initial post-stumble configuration $\tilde{q}_p$. For $i = 4, 5$ (i.e. the prosthetic joint angles), we write each $J_i(t) \subset \mathbb{R}^4$ as

$$J_i(t) = (x_i(t), \{g_i^k(t), g_i^j(t)\}, \{\langle \kappa_i \rangle, \langle \beta_i^j(t) \rangle\})^{p(t)}, \tag{5.15}$$

where $x_i(t) = [c_i^x, s_i^x, 0.5]^\top$, $g_i^k(t) = [c_i^k, s_i^k, 0.5]^\top$, and $g_i^j(t) = [c_i^j, s_i^j, 0]^\top$ for $j = 1, ..., p(t)$. We created JRSs which start from $q_i(0; k) = 0$ as in Def. 5.3.6. However, in general the stumble may start at any phase $t_s \in [0, 1]$. Let q_i^{offset} be the angle corresponding to the cosine and sine dimensions of $x_i(t_s)$, i.e. the center of the first post-stumble JRS zonotope. Then, we have

$$\begin{aligned} M_i(t) &= R_i(\tilde{q}_i) R_i^{-1}(q_i^{\text{offset}}) \left(X_i(t), \left\{ G_i^k(t), G_i^j(t) \right\}, \left\{ \langle \kappa_i \rangle, \langle \beta_i^j(t) \rangle \right\} \right)^{p(t)}, \\ X_i(t) &= I_{3\times 3} + s_i^x U_i + (1 - c_i^x) U_i^2, \\ G_i^k(t) &= s_i^k U_i - c_i^k U_i^2 \\ G_i^j(t) &= s_i^j U_i - c_i^j U_i^2 \end{aligned} \tag{5.16}$$

for $i = 4, 5$. The rotation by $\tilde{q}_i$ and inverse rotation by q_i^{offset} ensure that the matrix zonotopes built from the JRSs correspond to the correct initial configuration. We call this the `makeProsMatZono` function in Alg. 4. Note that these matrix zonotopes are guaranteed to overapproximate rotations induced by the trajectory parameters; see Lem. 4.4.9 for details.

Once the matrix zonotopes for all of the hip and prosthetic joints have been formed, we compute their products to overapproximate the forward occupancy maps FO_{heel} and FO_{toe} at each time step. Here, we use the rotatotopes $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ overapproximate $\mathscr{L}_{\text{heel}}$ and $\mathscr{L}_{\text{toe}}$. For information about rotatotopes, please refer to Sec. 4.4.1. Recalling that l_{th}, l_{sk}, l_{hl} and l_{toe} define the necessary

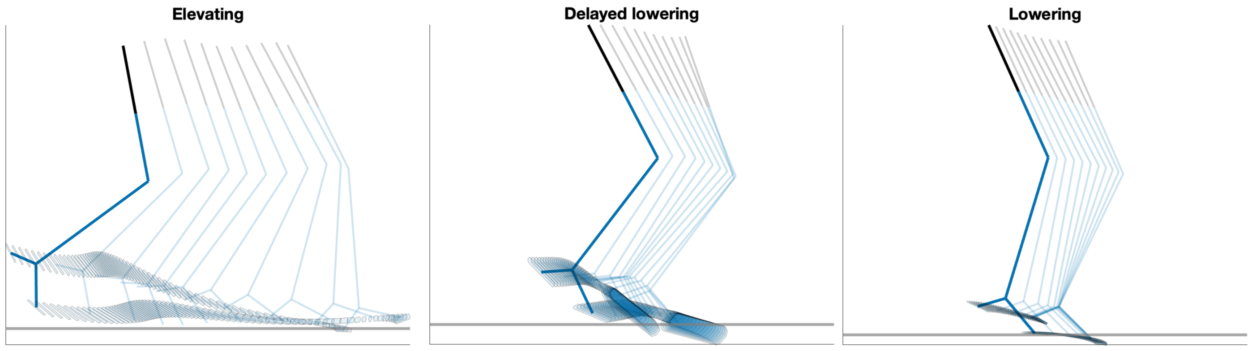


Figure 5.7: This figure displays example rotatotopes $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ for each of the three trip recovery strategies for Subject ID 1. The stick figures show the observed motion of the subject's swing leg following a perturbation until contact with the ground for a single trial. The observed hip motion in this trial was treated as a "prediction", then combined with the parameterized JRSs to construct $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ as in (5.17) and (5.18). These rotatotopes, shown as the shaded blue polygons, are defined using the correct initial configuration $\tilde{q}_p$ at the onset of stumble $t = t_s$. They overapproximate the true motion of the heel and toe trajectories when following the trajectory parameterization shown in Fig. 5.4 and Fig. 5.5. In Sec. 5.5.3, we use these rotatotopes to generate foot placement and clearance constraints which ensure a safe recovery.

lengths of our kinematic model, we may treat these as zonotopes with no generators. Then, we compute $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ by multiplying the matrix zonotopes $M_i(t)$ by these lengths in the order that they appear within the forward occupancy maps FO_{heel} and FO_{toe}. These rotatotopes may then be shifted by the set of possible hip positions, given by $J_{\text{xyz}}(t)$, by a Minkowski sum. These operations are implemented in Alg. 4 and formalized below:

Lemma 5.5.2. *For any $t \in [t_s, 1]$ and $k \in K$, $\text{FO}_{\text{heel}}(q(t;k)) \subseteq V_{\text{heel}}(t)$ and $\text{FO}_{\text{toe}}(q(t;k)) \subseteq V_{\text{toe}}(t)$, where*

$$V_{\text{heel}}(t) = J_{\text{xyz}}(t) \oplus \prod_{i=1}^{3} M_i(t) l_{\text{th}} \oplus \prod_{i=1}^{4} M_i(t) l_{\text{sk}} \oplus \prod_{i=1}^{5} M_i(t) R(\theta_{\text{heel}}) l_{\text{hl}} \tag{5.17}$$

$$V_{\text{toe}}(t) = J_{\text{xyz}}(t) \oplus \prod_{i=1}^{3} M_i(t) l_{\text{th}} \oplus \prod_{i=1}^{4} M_i(t) l_{\text{sk}} \oplus \prod_{i=1}^{5} M_i(t) l_{\text{toe}} \tag{5.18}$$

Upon close inspection, these equations are identical to the forward occupancy equations found in (5.3) and (5.4), where we have replaced all matrices with matrix zonotopes and replaced the hip position with a set of possible positions. Lemma 5.5.2 lets us overapproximate $\mathscr{L}_{\text{heel}}$ and $\mathscr{L}_{\text{toe}}$, i.e. $\mathscr{L}_{\text{heel}} \subseteq \bigcup_{t\in[t_s,1]} V_{\text{heel}}(t)$ and $\mathscr{L}_{\text{toe}} \subseteq \bigcup_{t\in[t_s,1]} V_{\text{toe}}(t)$. An example of the computed rotatotopes $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ are displayed in Fig. 5.7.

As in Chap. 4, the critical point to remember about $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ is that they maintain the trajectory parameterization by $k \in K$. That is, even though $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ are in the workspace, we may *slice* them by the trajectory parameter $k = (k_4, k_5)$ to obtain subsets of $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ corresponding to specific motions of the prosthetic joints. In the next subsection, we again only consider generators which are fully-k-sliceable, which means that all of their indeterminates are evaluated by a particular choice of trajectory parameter. For more context on slicing and the evaluation of indeterminates, see Alg. 1.

5.5.3 Constraint Generation

5.5.3.1 Theory

The goal of this work is to plan and execute trajectories of a prosthesis that robustly respond to stumbles that are sensed online. Importantly, we want to choose trajectory parameters k which lead to successful recovery across a range of plausible human trajectories. We formulate this as an optimization problem, with constraints ensuring that the reachable set of swing heel/toe positions at touchdown corresponding to k is contained within the target set X_T. Furthermore, depending on the trip recovery strategy, we want to prevent the prosthetic heel/toe from catching on the ground during swing. To be clear about the strategy-dependent role of the heel and toe in these placement and clearance constraints, we let $\mathscr{L}_{\text{placement}} = \mathscr{L}_{\text{heel}}$ and $\mathscr{L}_{\text{clearance}} = \mathscr{L}_{\text{toe}}$ for the *elevating* strategy, and let $\mathscr{L}_{\text{placement}} = \mathscr{L}_{\text{toe}}$ and $\mathscr{L}_{\text{clearance}} = \mathscr{L}_{\text{heel}}$ for the *delayed lowering* and *lowering* strategies. Furthermore, because the foot contact time t_f depends on both the reachable sets and choice of trajectory parameters, we let $t_{f,\text{max}}(k)$ be the last possible contact time for a given choice of trajectory parameters. Given $\mathscr{L}_{\text{placement}}$ and $\mathscr{L}_{\text{clearance}}$, let K_u be the set of *unsafe* trajectory parameters for the prosthesis which is defined as follows.

$$K_{\text{timing}} = \{k \in K \mid t_{f,\text{max}}(k) > 1\} \tag{5.19}$$

$$K_{\text{clearance}} = \{k \in K \mid (X, k, t) \subset \mathscr{L}_{\text{clearance}}, t \leq t_{f,\text{max}}(k), \exists z \in \pi_z(X) \text{ s.t. } z \leq 0\} \tag{5.20}$$

$$\begin{aligned} K_{\text{placement}} = \{k \in K \mid (X, k, t) \subset \mathscr{L}_{\text{placement}}, (Y, t) \subset \mathscr{J}_{\text{COM}}, t \leq t_{f,\text{max}}(k), \ldots \\ \exists z \in \pi_z(X) \text{ s.t. } z \leq 0,\ \exists (x, y)^\top \in \pi_{xy}(X) \text{ s.t. } (x, y)^\top \notin \pi_{xy}(X_T \cap Y)\} \end{aligned} \tag{5.21}$$

$$K_u = K_{\text{timing}} \cup K_{\text{clearance}} \cup K_{\text{placement}}. \tag{5.22}$$

In short, $K_{\text{clearance}}$ says that a trajectory parameter k is unsafe if the clearance reachable set can intersect the ground at any time *prior to contact* by the desired portion of the foot. $K_{\text{placement}}$ says that k is unsafe if any time prior to swing foot contact that the placement reachable set intersects the ground, its projection onto the ground plane is not fully contained within the target set X_T intersected with all predicted COM states. In this context, we let $\pi_{xy}(X_T \cap Y)$ be a projection onto the dimensions of X_T representing foot placement positions on the ground plane. The definition of $K_{\text{placement}}$ reflects the fact that the contact time t_f depends on both the reachable sets and choice of trajectory parameters. K_{timing} ensures that contact occurs before the maximum allowable phase.

5.5.3.2 Implementation

We overapproximate $K_{\text{clearance}}, K_{\text{placement}}$ and K_{timing} using constraint functions $h_{\text{clearance}}, h_{\text{placement}}$ and h_{timing}.

As in Sec. 4.4.3.2, we form these constraints by considering *slices* of the rotatotopes $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ corresponding to particular choices of k. Here, we again only consider generators of $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ that are *fully-k-sliceable*, and treat all other generators conservatively. For more information about these concepts, we refer to Sec. 4.4.1. As before, we let $V_{\text{placement}}(t) = V_{\text{heel}}(t)$ and $V_{\text{clearance}}(t) = V_{\text{toe}}(t)$ for the *elevating* strategy, and let $V_{\text{placement}}(t) = V_{\text{toe}}(t)$ and $V_{\text{clearance}}(t) = V_{\text{heel}}(t)$ for the *delayed lowering* and *lowering* strategies.

First, we explain our representation of K_{timing}. Consider a particular choice of k. We want to ensure that the corresponding subset of a rotatotope $V_{\text{placement}}(t)$ is completely below the ground at some phase $t < 1$. To check this, we separate $V_{\text{placement}}(t)$ into two rotatotopes (similarly to Sec. 4.4.3.2, (4.32)). The first will contain only fully-k-sliceable generators, while the second will be treated a zonotope and used to "buffer" for overapproximation:

$$V_{\text{placement, slc}}(t) = \left(x_{\text{placement}}(t), g^j_{\text{slc}}, \langle \kappa^j_{\text{slc}} \rangle\right) \text{ and } V_{\text{placement, buf}}(t) = (0, g^n_{\text{buf}}, \langle \beta^n_{\text{buf}} \rangle)\,, \tag{5.23}$$

such that $V_{\text{placement}}(t) = V_{\text{placement, slc}}(t) \oplus V_{\text{placement, buf}}(t)$, where $V_{\text{placement, slc}}(t)$ has only fully-k-sliceable generators. That is, each $\langle \kappa^j_{\text{slc}} \rangle$ is a product of *only* $\langle \kappa^4 \rangle$ and $\langle \kappa^5 \rangle$. Note, the number of generators/indeterminates in $V_{\text{placement, slc}}(t)$ and $V_{\text{placement, buf}}(t)$ is omitted to ease notation.

For any $k \in K$, slicing $V_{\text{placement, slc}}(t)$ by k returns a single point in $\mathbb{R}^3$. We express this with $\texttt{eval} : \mathcal{P}(W) \times K \to \mathbb{R}^3$ for which

$$\texttt{eval}(V_{\text{placement, slc}}(t), k) = \texttt{slice}\left(V_{\text{placement, slc}}(t), \left\{\langle \kappa^4 \rangle, \langle \kappa^5 \rangle\right\}, \{2k_4 - 1, 2k_5 - 1\}\right) \tag{5.24}$$

where the values $2k_4 - 1$ and $2k_5 - 1$ map $k \in K = [0,1] \times [0,1]$ to corresponding values of generator coefficients between $[-1, 1]$. Note, `eval` can be implemented as the evaluation of polynomials and the `slice` algorithm is given in Alg. 1.

Let $\pi_z(V_{\text{placement, buf}}(t))$ be the projection of $V_{\text{placement, buf}}(t)$ onto the vertical axis. Recall that we assume level-ground walking, and that the ground plane has height $z = 0$. Notice that $\pi_z(V_{\text{placement, buf}}(t)) \subset \mathbb{R}$ is an interval of heights. We define a function $h_{z,\text{max}} : [0,1] \times K \to \mathbb{R}$

$$h_{z,\text{max}}(t, k) = \pi_z(\texttt{eval}(V_{\text{placement, slc}}(t), k)) + \max(\pi_z(V_{\text{placement, buf}}(t))) \tag{5.25}$$

where $h_{z,\text{max}}(t, k) < 0$ for a given t and k implies that the sliced rotatotope is completely *below* ground. Then, $t_{f,\text{max}}(k)$ can be found as the first phase t where $h_{z,\text{max}}(t, k) < 0$. We write h_{timing} as

$$h_{\text{timing}}(k) = t_{f,\text{max}}(k) - 1 \tag{5.26}$$

where $h_{\text{timing}}(k) < 0$ implies a foot contact occurs before the maximum phase of $t = 1$.

Lemma 5.5.3. *If $k \in K_{\text{timing}}$, then $h_{\text{timing}}(k) \geq 0$.*

We define a similar function $h_{z,\text{min}} : [0, 1] \times K \to \mathbb{R}$

$$h_{z,\text{min}}(t, k) = \pi_z(\texttt{eval}(V_{\text{placement, slc}}(t), k)) + \min(\pi_z(V_{\text{placement, buf}}(t))) \tag{5.27}$$

where $h_{z,\text{min}}(t, k) > 0$ for a given t and k implies that the sliced rotatotope is completely *above* ground.

Next, we consider a representation of $K_{\text{placement}}$. We want to ensure that whenever the foot can make contact before the ground (up until the maximum phase $t_{f,\text{max}}$), the foot placement location lies within the target set X_T (Def. 5.3.8). This means that although we do not know exactly when the heel or toe will make contact with the ground, we can guarantee that it will be placed appropriately, as described by X_T. Notice that contact with the ground is possible for a given t and k whenever $h_{z,\text{min}}(t, k) \leq 0$.

At a given phase t, let $Y(t)$ represent all predicted reachable COM states of the system. That is, $Y(t) = \{y \,|\, (y, t) \in \mathscr{J}_{\text{COM}}\}$. $Y(t)$ determines the subset of X_T under consideration, which limits possible foot placement positions. In particular, if the foot contacts the ground at phase t, we require that the foot be placed in $\pi_{xy}(X_T \cap Y(t))$, which may be represented as a polytope.

We do not know exactly where the foot will be placed, due to the overapproximative method for constructing reachable sets. To guarantee safety, we have to ensure that *all possible* foot placement positions satisfy the target set constraint. To implement this, we require

$$\pi_{xy}(\texttt{eval}(V_{\text{placement, slc}}(t), k) \oplus V_{\text{placement, buf}}(t)) \subseteq \pi_{xy}(X_T \cap Y(t)). \tag{5.28}$$

In practice, we take the Minkowski (or Pontryagin) difference [225] of $\pi_{xy}(X_T \cap Y(t))$ and $\pi_{xy}(V_{\text{heel, buf}}(t))$, which yields a *constraint polytope X_C*:

$$X_C = \{x \in \mathbb{R}^2 \mid x \oplus \pi_{xy}(V_{\text{placement, buf}}(t)) \subseteq \pi_{xy}(X_T \cap Y(t))\}. \tag{5.29}$$

Let (A, b) be the halfspace representation of the constraint polytope, i.e. $x \in X_C \iff Ax - b < 0$. Then, $h_{\text{placement}}$ can be written as

$$h_{\text{placement}}(t, k) = \min(-h_{z,\text{min}}(t, k), \max(A\pi_{xy}(\texttt{eval}(V_{\text{placement, slc}}(t), k)) - b)). \tag{5.30}$$

First, notice that $h_{z,\text{min}}(t, k) \leq 0$ implies that a ground contact is possible, which is equivalent to $-h_{z,\text{min}}(t, k) \geq 0$. Also, notice that $\max(A\pi_{xy}(\texttt{eval}(V_{\text{placement, slc}}(t), k)) - b) \geq 0$ implies that the point $\pi_{xy}(\texttt{eval}(V_{\text{placement, slc}}(t), k)$ is not in the interior of the constraint polytope defined by (A, b). Therefore, we consider it unsafe if both ground contact is possible and the foot placement is not robust, i.e. both $-h_{z,\text{min}}(t, k) \geq 0$ and $\max(A\pi_{xy}(\texttt{eval}(V_{\text{placement, slc}}(t), k)) - b) \geq 0$. By taking the minimum of these two values, the condition $h_{\text{placement}}(t, k) < 0$ enforces that the foot is placed appropriately whenever it can possibly touch the ground. This is demonstrated in Fig. 5.8.

Lemma 5.5.4. *If $k \in K_{\text{placement}}$, then there exists $t \leq t_{f,\text{max}}(k)$ such that $h_{\text{placement}}(t, k) \geq 0$.*

Finally, we discuss our representation of $K_{\text{clearance}}$, which identifies trajectory parameters which may cause unwanted collision between the foot and the ground. We break $V_{\text{clearance}}(t)$ into components $V_{\text{clearance, slc}}(t)$ and $V_{\text{clearance, buf}}(t)$ just as we did for the placement rotatotopes in (5.23). We define a similar function for the clearance height, $h_{z,\text{clearance,min}} : [0, 1] \times K \to \mathbb{R}$

$$h_{z,\text{clearance,min}}(t, k) = \pi_z(\texttt{eval}(V_{\text{clearance, slc}}(t), k)) + \min(\pi_z(V_{\text{clearance, buf}}(t))) \tag{5.31}$$

where $h_{z,\text{clearance,min}}(t, k) \leq 0$ for a given t and k implies that the sliced rotatotope *may* intersect with the ground. Therefore, letting

$$h_{\text{clearance}}(t, k) = -h_{z,\text{clearance,min}}(t, k), \tag{5.32}$$

we can ensure that the sliced clearance rotatotopes never intersect with the ground.

Lemma 5.5.5. *If $k \in K_{\text{clearance}}$, then there exists $t \leq t_{f,\text{max}}(k)$ such that $h_{\text{clearance}}(t, k) \geq 0$.*

5.5.4 Online Trajectory Optimization

5.5.4.1 Theory

Letting $\phi(k)$ be a user-specified cost function, we solve for the optimal trajectory parameter k as

$$k_{\text{opt}} = \text{argmin}_{k \in K}\{\phi(k) \mid k \notin K_u\}. \tag{5.33}$$

For example, $\phi(k)$ may penalize the distance from a desired trajectory, or the work done by the prosthesis. As in (5.19), $K_u = K_{\text{timing}} \cup K_{\text{clearance}} \cup K_{\text{placement}}$. Unfortunately, the optimization (5.33) may not be feasible if every trajectory parameter is unsafe, i.e. $K = K_u$. In this case, safety can not be ensured. TRIP-RTD defaults to using the nominal trajectory in this case, but in subsequent planning iterations the problem may become feasible.

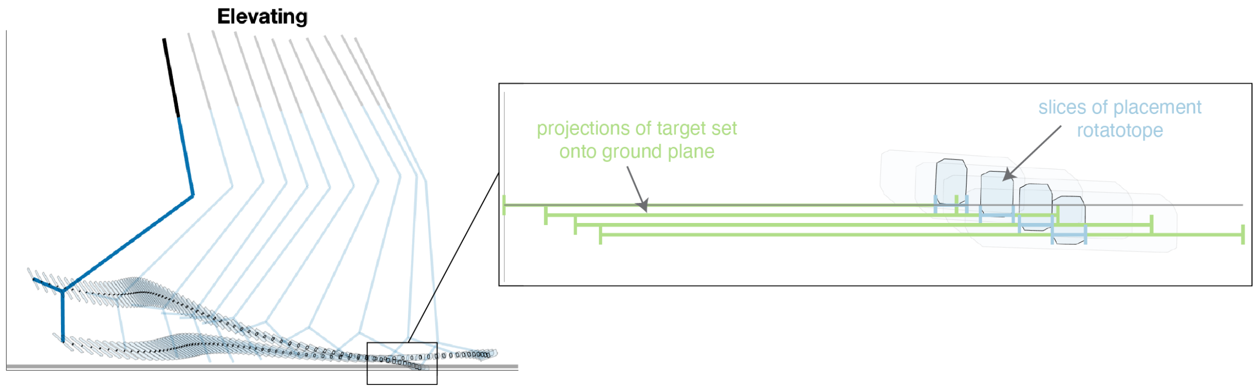

Elevating
projections of target set onto ground plane
slices of placement rotatotope

Figure 5.8: This figure shows projections of the target set onto the ground plane (light green intervals) for the phase indexes where contact between $V_{\text{heel}}(t)$ and the ground are possible for the elevating example shown in Fig. 5.7. Because these projections depend on the motion of the person's COM, notice that they progress forwards as swing progresses because of the forward progression of the COM. We plot the rotatotopes $V_{\text{heel}}(t)$ (right side, faint blue shapes) at the phase indexes where contact with the ground is possible. In particular, slicing $V_{\text{heel}}(t)$ by a particular k yields subsets of these rotatotopes (right side, filled blue shapes). These subsets are either completely above ground or completely contained within the target set at each phase index and therefore satisfy the placement constraint in (5.35). In this instance, the leftmost slice is completely above ground, and subsequent slices fall within the target set (light blue intervals).

Assuming the problem (5.33) was feasible and an optimal trajectory parameter has been identified, the control input for the prosthetic knee at time t is given by a tracking controller

$$u_p(t) = G_1\big(q_p(t) - q_{p,\text{des}}(t;k)\big) + G_2\big(\dot{q}_p(t) - \dot{q}_{p,\text{des}}(t;k)\big), \tag{5.34}$$

where G_1 and G_2 are user specified feedback gains, and we emphasize that the trajectory parameters produce desired joint trajectories to be tracked by the feedback controller.

5.5.4.2 Implementation

We implement (5.33) as a nonlinear program, denoted `optTraj` in Alg. 5:

$$\begin{aligned} k_{\text{opt}} = \underset{k\in K}{\text{argmin}} \quad & \phi(k) \\ \text{s.t.} \quad & h_{\text{timing}}(k) < 0 \\ & h_{\text{clearance}}(t,k) < 0 \quad \forall\, t \in [t_s, t_{f,\max}(k)] \\ & h_{\text{placement}}(t,k) < 0 \quad \forall\, t \in [t_s, t_{f,\max}(k)]. \end{aligned} \tag{5.35}$$

Theorem 5.5.6. *Any feasible solution k to* (5.35) *parameterizes a strategy-specific trajectory which*

Algorithm 5 $q_{\text{plan}} = \texttt{makePlan}(\text{strategy}, \tilde{q}_p, \dot{\tilde{q}}_p, q_{p,\text{nom}}, \phi, \mathscr{J}_{\text{xyz}}, \mathscr{J}_i(i = 1, 2, 3), \mathscr{J}_{\text{COM}})$

```
1: $\{V_{\text{heel}}(t), V_{\text{toe}}(t)\} \leftarrow \texttt{composeRS}(\tilde{q}, \dot{\tilde{q}})$ // Sec. 5.5.2
2: $(h_{\text{timing}}, h_{\text{placement}}, h_{\text{clearance}}) \leftarrow \texttt{makeCons}(\text{strategy}, \tilde{q}, \dot{\tilde{q}}, \mathscr{O}, \{V_i(t)\})$ // Sec. 5.5.3
3: // solve (5.35) within $t_{\text{plan}}$ or else return $q_{p,\text{nom}}$.
4: $q_{\text{plan}} \leftarrow \texttt{optTraj}\left(\phi, h_{\text{timing}}, h_{\text{placement}}, h_{\text{clearance}}, t_{\text{plan}}, q_{p,\text{nom}}\right)$ // Sec. 5.5.4
```

makes contact with the ground before $t_{f,\max}(k) < 1$*, places the prosthetic foot robustly with respect to the target set, and enforces clearance constraints which avoid unwanted foot contact across all predicted motions of the hip and COM.*

TRIP-RTD uses Alg. 5 in each planning iteration. Each plan lasts for the phase horizon $[t_s, 1]$, where the nominal prosthesis stance controller is assumed to resume control upon contact with the ground. TRIP-RTD correctly accounts for the initial position of the prosthetic leg at the time of the stumble, and utilizes reachable sets of both human and prosthetic subsystems to identify safety constraints. TRIP-RTD attempts to find a safe trajectory within t_{plan} subject to these safety constraints. However, if no safe trajectory can be found, TRIP-RTD defaults to using the nominal phase-based trajectory $q_{p,\text{nom}}$.

5.6 Demonstrations

5.6.1 Simulation Experiments on Preexisting Tripping Dataset

We test TRIP-RTD in simulation using the tripping dataset described in Sec. 5.2, which was collected during a previous experiment [3] and graciously made accessible to us. Two able-bodied subjects' kinematic data were used in our simulation experiment; additional subjects' data from this dataset requires further processing before use. Annotations in the dataset give the timing of tether-trips and events such as heel strikes and toe offs. For this simulation experiment, we replace the subjects' actual knee and ankle motion by a simulated prosthetic leg which is controlled by TRIP-RTD. We hypothesize that TRIP-RTD can produce trajectories of the simulated prosthesis which prevent falls in amputees and lead to successful walking following a trip.

The testing procedure proceeds as follows. For each subject, we assume bilateral symmetry and treat left and right swing phases identically. The link lengths of the kinematic model (shown in Fig. 5.3) are fit to observed joint positions. Joint angles are extracted from the observed kinematics. We approximate the COM position using the subject's pelvis. Perturbed swing phases are heuristically sorted into each trip-recovery strategy as in [3]. If a perturbation results in an "incomplete arrest" – the forward motion of the foot is not stopped by the tether – it is excluded from testing.

Nominal trajectories of the subjects' knee and ankle motion are fit to unperturbed swing phases.

The trajectory parameterization and target sets are fit to the observed trajectories for each strategy as described in Sec. 5.3.4.2 and Sec. 5.3.5.2. We note that we test TRIP-RTD on the same data that we use to fit the trajectory parameterization and targets sets. We avoid overfitting the trajectory parameters by using a low-dimensional linear parameterization by $k \in K$, but future studies should avoid training and testing on the same data.

As described in Sec. 5.5.1, ground-truth trajectories of each subject's hip and COM motion are treated as "predictions" and passed to TRIP-RTD. These predictions span from the onset of the tripping perturbation until the swing foot contacts the ground. In practice, the predictions should be zonotopes which overapproximate the continuous hip and COM trajectories. However, for this study we create zonotopes representing the predictions at discrete time steps, which results in potential gaps between time steps for $V_{\text{heel}}(t)$ and $V_{\text{toe}}(t)$ which can be seen in close examination of Fig. 5.7.

We briefly cover several simplifying details used for this experiment before presenting the results. TRIP-RTD can plan in a receding-horizon fashion, as described in Sec. 5.3.3. However, for the results presented here we take a more restrictive view and only allow a single post-stumble planning iteration. Although TRIP-RTD is presented in 3D throughout this chapter, for this experiment we assume all motion takes place in the sagittal plane and test in 2D. Lastly, we elect not to enforce the timing constraint found in (5.35). Because our "predictions" only run until the foot makes contact with the ground, it is impossible to say what the trajectories of the hip and COM would have been beyond this point. Therefore, implementing this constraint would require TRIP-RTD to always make contact with the ground *before* the subject did.

In this experiment, we are only interested in TRIP-RTD finding feasible solutions to (5.35) which guarantee a safe trip recovery. Therefore, we set the cost function $\phi(k) = 0$. We compare TRIP-RTD to a baseline nominal controller. Specifically, we examine whether the nominal position trajectories of the knee and ankle – coupled with the observed hip motion – would satisfy the clearance and placement constraints in (5.35). If they do, we say that the nominal controller produces a feasible trajectory. The results of this experiment are given in Tab. 5.1.

Overall, TRIP-RTD finds a feasible trajectory in 83/112 trials, or 74.11% of the time. In contrast, the nominal position trajectories were feasible in only 8/112 trials, 7.14% of the time. We note that we construct the nominal trajectory using a time-based phase variable in Sec. 5.3.2.2. The use of a different phase variable arising from kinematic data may allow the nominal trajectory to perform significantly better. However, because TRIP-RTD returns the nominal trajectory if no feasible trajectory is found, it can only improve on the nominal controller for fall prevention. These results demonstrate that TRIP-RTD may supplement existing prosthesis control methods for the task of trip recovery.

For subject ID 2, no elevating trials are observed, and for subject ID 1, only 6 lowering trials oc-

Subject	Controller	Elevating	Delayed Lowering	Lowering
ID 1	TRIP-RTD	16/22 – 72.73%	15/20 – 75.00%	2/6 – 33.33%
	Nominal	0/22 – 0.00%	0/20 – 0.00%	1/6 – 16.67%
ID 2	TRIP-RTD	0/0	35/46 – 76.09%	15/18 – 83.33%
	Nominal	0/0	0/46 – 0.00%	7/18 – 38.89%
Combined	TRIP-RTD	16/22 – 72.73%	50/66 – 75.76%	17/24 – 70.83%
	Nominal	0/22 – 0.00%	0/66 – 0.00%	8/24 – 33.33%

Table 5.1: The results of the simulation study for subjects ID 1 and 2. We present the number of trials where feasible trajectories were found relative to the number of trials observed for each strategy.

curred. The low feasibility rate for subject ID 1's lowering strategy trials can be partially explained by this low number of trials, where it may be difficult to fit an adequate trajectory parameterization to data.

We note that because we removed the timing constraint from (4.40) for this experiment, some of the feasible trajectories may not make contact with the ground before the maximum phase $t = 1$. This happens in 28/83 (33.73%) feasible TRIP-RTD trajectories, and 8/8 (100%) feasible nominal trajectories. Without predictions of the hip and COM that extend beyond the observed foot contact with the ground, it is difficult to speculate about whether these trajectories would still be feasible in real-world applications.

TRIP-RTD was implemented in MATLAB, and (4.40) was solved using `fmincon` from a random initial condition. The constraint generation and trajectory optimization combined take 94 ms on average when a feasible trajectory is found. However, in the present formulation, the online generation of the reachable sets takes 1 to 2 seconds. As in ARMTD (Chap. 4), this time can be significantly reduced by a faster implementation (i.e. using C++) and parallelizing across time steps.

5.7 Conclusion

Prosthetic stumble-recovery controllers could help people with major lower-limb loss perform everyday tasks with more confidence. However, the uncertainty in measuring and estimating human behavior have made designing these controllers difficult. In this work, we present a method for planning stumble-recovery trajectories online while explicitly accounting for a range of predicted human responses. Our method constructs reachable sets that depend on both the human and prosthesis subsystems, then selects a trajectory parameter for the prosthetic knee and ankle online to ensure that the entire system recovers successfully. To our knowledge, this work represents the first prosthetic controller framework that can plan trajectories online while ensuring safety across a range of possible behaviors.

Recently, we received IRB approval to begin conducting our own experiment which replicates the tether-trip perturbations found in previous work. In the experiment, a tether attached to the prosthesis' foot will induce stumbles by temporarily halting the forward progress of the foot in swing phase. We plan to use the same evaluation procedure as described in the previous subsection for the first part of our own experiment. Subsequently, we will utilize TRIP-RTD with an open-source robotic leg [226]. We plan to have the same able-bodied persons studied in the first part of the experiment walk with this leg via a bypass adaptor. The goal is to test if the trajectory parameterizations and target sets built from each subject's data allows TRIP-RTD to generate successful trip recoveries using a prosthesis.

Getting TRIP-RTD to work successfully on hardware requires overcoming a number of difficult challenges. First, in this chapter we assume that conservative predictions of future hip and COM motion are given. Generating these predictions online is extremely challenging. Current methods for classifying trip recovery strategies [221] and generating estimates of trajectories [219] have made promising strides towards achieving this. Presently, we are investigating the use of Gaussian processes to predict these trajectories online, similarly to [219]. With this model we can set a threshold from the mean (such as two standard deviations), and form zonotopes which overapproximate these subsets of the distribution to generate set-based predictions.

Second, using TRIP-RTD with only a computer onboard a prosthesis may be computationally challenging. For example, ARMTD in Chap. 4 leveraged a GPU to enable real-time computation by parallelizing many of the steps of the algorithm. This option may not be possible for TRIP-RTD given the limited computational resources onboard a robotic leg.

Third, TRIP-RTD may be overly conservative by requiring a plan to be safe across all predicted hip and COM motion. In practice, it may be more useful to try to maximize the probability of a successful recovery, given distributions which represent the likelihood of specific human actions. However, propagating distributions of hip angles through the kinematic chain to get positions of the prosthetic foot is an open problem.

Fourth, in this chapter we relied on able-bodied subjects' data to form the trajectory parameterizations and target sets for TRIP-RTD. For amputees, such data may not be available. Work must be done to generalize the parameterizations and target sets so that TRIP-RTD can be transferred to other persons without requiring extensive individual data.

The TRIP-RTD framework can be generalized to other prosthesis tasks by developing (1) sets of predicted human behavior in other environments and (2) a broader range of parameterized prosthesis behavior. Examples of potential applications include avoiding obstacles, ascending/descending stairs, and maintaining balance on narrow paths. Furthermore, TRIP-RTD can be adapted for use in other wearable robots, like powered exoskeletons and soft exosuits. TRIP-RTD provides guarantees that the trajectories it produces lead to successful recovery from trips. This important property of safe autonomy during stumble recovery can help assistive robots reduce the incidence of falls, while ensuring that recovery actions taken by robots will not cause a fall by seeking to prevent a fall.

CHAPTER 6

Conclusion and Future Directions

Assistive robots will play a vital role in maintaining and improving quality of life for older persons and persons with disabilities in the coming decades. These robots have the capacity to assist with activities of daily living, reducing the load of caregivers and increasing access to affordable care. To achieve these goals, these robots must be **safe**. This dissertation advances the current state of the art for safe autonomy in assistive robots by making contributions to the interconnected domains of perception, monitoring, manipulation and fall prevention. The approaches in this dissertation focus on certificates of safety and optimality to increase trust in these algorithms when employed in unstructured, uncertain environments. Specifically, we give theoretical guarantees for each of the tools presented in this dissertation in Sec. 2.3.4 (perception), Thm. 3.6.2 (monitoring instability during motion), Thm. 4.4.15 (collision-free manipulation), and Thm. 5.5.6 (fall prevention).

Though distributed across separate chapters, the goal of this dissertation is to provide a unified framework for safe autonomy in assistive robots. First, accurate perception (Chap. 2) is critical for localizing humans in the environment. These perceptual estimates may be passed to predictive algorithms, which generate estimates of likely future actions. By understanding where a person is likely to be, manipulators operating near humans can guarantee collision avoidance by treating humans as obstacles (Chap. 4). Perceptual estimates may also be used to monitor a person's biomechanical health over time and characterize their stability. Such monitoring algorithms can use models of a person's motion to reduce the likelihood of falling by alerting robots to impending user instability (Chap. 3). Wearable robots, like powered prostheses, may then plan appropriate recovery strategies that prevent falls across a set of predicted human responses (Chap. 5).

In this chapter we provide a summary of the contributions described in this dissertation, as well as future research directions to build on the present contributions.

6.1 Discussion of Contributions

Chapter 2, Perception of 3D Pose Chapter 2 presents an optimization method for estimating 3D pose from multi-view 2D pose estimates which provides certificates of the optimality of its solutions. The method is based on the Sparse Bounded-degree Sums-of-Squares hierarchy. We demonstrate that the method achieves or exceeds the accuracy of current state-of-the-art methods which use neural networks to estimate 3D pose. At the same time, the method takes less time to make inferences, incorporates strict link length constraints, and provides attractive theoretical guarantees on the optimality of its solutions. Importantly, the method requires no 3D training data. This advantage improves its generalizability, which we demonstrate by producing accurate 3D results when utilizing a generic 2D pose detector that was trained only on 2D images from a different dataset.

Chapter 3, Monitoring of Instability Chapter 3 describes the use of reachable sets for characterizing human stability and demonstrates the efficacy of this approach via a sit-to-stand experiment where subjects were perturbed to instability by motor-driven cable pulls. The method utilizes kinematic observations of a subject performing a task to construct individualized dynamic and controller models that account for strategy-specific behavior. These models are used to describe the evolution of trajectories that end in a target set that characterizes successful completion of the task. The stable regions of motion, which we call Stability Basins, are computed as the backwards reachable sets of the target sets under the models' dynamics. We show that Stability Basins accurately predict a subject's instability (leading to stepping or sitting) before the step or sit is initiated, and also that they accurately predict successful completion of the sit-to-stand task. These results are demonstrated across 11 young subjects without disabilities performing three distinct sit-to-stand strategies. The method for computing Stability Basins requires no observations of failure, which makes it a promising method for quantifying stable motion via sensor-based observation in environments like nursing homes, where perturbing a subject to failure could be impractically dangerous.

Chapter 4, Planning for Manipulators Chapter 4 introduces ARMTD, a guaranteed-safe receding-horizon trajectory planner for manipulators. ARMTD quickly composes parameterized reachable sets of trajectories of the arm's volume through workspace and finds plans that guarantee the arm can not collide with the environment or itself. The method improves on current state-of-the-art approaches by tractably constructing these reachable sets online from low dimensional Joint Reachable Sets, which describe the rotation of each joint. ARMTD correctly overapproximates to account for continuous-time trajectories, and contains a fail-safe maneuver in each receding-horizon iteration to ensure that a safe trajectory is always available. The method is being extended to account for kinodynamic uncertainty, which arises when the inertial properties of the arm or an object in its grasp are not perfectly known. ARMTD is tested extensively in simulation, where it never crashes, as well as on hardware with a Fetch Mobile Manipulator robot, where it can plan around realistic obstacles and account for changing environments. The algorithm is fully parallelized and implemented on a GPU to achieve real-time performance.

Chapter 5, Fall Prevention via Prosthetic Trip Recovery Chapter 5 proposes TRIP-RTD for planning trip recoveries for powered prosthetic legs when a trip occurs while the prosthesis is in swing. After a trip is detected, TRIP-RTD loads a stumble-recovery library based on the predicted desired recovery strategy, which can be classified as elevating, delayed lowering, or lowering [220]. TRIP-RTD builds on the ARMTD framework, and utilizes Joint Reachable Sets of prosthetic knee and ankle motion for each stumble-recovery strategy to compose parameterized reachable sets of the prosthesis' heel and toe through the workspace. Depending on predictions of the person's hip and COM motion, TRIP-RTD selects a trajectory that ensures the swing foot is placed at an appropriate time and place to start the next step. TRIP-RTD is able to incorporate set-based predictions of human motion and guarantees safety across sets of potential trajectories, instead of the single most likely one. TRIP-RTD shows promise for preventing falls and found a guaranteed-safe trajectory for 74% of observed trips in simulation experiments with able-bodied human data.

6.2 Future Directions

While these contributions represent steps *towards* safe autonomy in assistive robots, much work must be done before the type of autonomy envisioned by this dissertation is actualized in practice. For example, this dissertation focuses explicitly on the "physical" safety of assistive robots, but ensuring the "psychological" safety of these robots (e.g. actions that build trust and social rapport) is an interesting and important subfield [61]. While this dissertation does not explicitly examine trust, the safety guarantees developed herein may be a avenue for building trust. Furthermore, building intelligent autonomy into assistive robots is challenging. ARMTD can guarantee the safety of its plans which achieve high-level goals, for example, but it makes no assumptions about how these high-level goals are generated. To be useful over long periods, assistive robots must intelligently specify high-level plans that assist a user without the intervention of human operators. It may be possible for a robot to continually learn how best to support a person, e.g. through safe reinforcement learning [227, 228]. Furthermore, steps must be taken to ensure the cyberphysical safety of these systems from potential hackers and attackers [229, 230, 231]

In the context of the current contributions and the problems addressed by the dissertation, we suggest the following future avenues of research to extend the generalizability and performance of the tools we have described.

6.2.1 3D Pose

Realistic models of the human body can be achieved through the use of skinning and blend shapes [232]. Instead of using keypoints to represent 3D joint positions (as in Chap 2), such a model could be used to fit the shape or volume occupied by the human. Neural-network based methods have demonstrated the use of these models for monocular shape estimation [233]. Recently, an optimization-based method has shown similarly promising results [234]. This method also utilizes the sparsity of the kinematic model to speed up the optimization and can operate very quickly, but can not certify the global optimality of its solutions. It may be possible to adopt a similar model and cost function within the SBSOS framework, but doing so may require a reformulation to avoid the high-degree polynomials that result from chained rotations.

The cost function in Chap. 2 (2.14) was designed to recreate 3D pose estimates from 2D camera estimates. However, one may add measurements from additional sensor modalities to the problem by simply adding these terms to the cost function. It is preferable, for computational reasons, for these added terms to remain quadratic in the decision variables. Measurement residuals using data from a 3D LIDAR point cloud would satisy this property, for example. Similarly, IMUs attached to the subject's body could be used to link together frames when exploiting the temporal consistency of human motion. For this case, my colleagues initially demonstrated SBSOS's utility for solving SLAM problems [99], which is similar to the problem of estimating position and orientation trajectories with an attached IMU.

6.2.2 Stability Basins

Wearable robots, such as powered exoskeletons, can assist with the sit-to-stand motion. The high predictive accuracy of Stability Basins can be utilized by these robots in real time with trajectories of the user's center of mass to predict when a subject will become unstable. At these times, the robot may need to switch its control strategy to help the user recover by planning to take a step or to sit back down in the seat. Furthermore, Stability Basins may serve as constraints for online trajectory optimization by requiring trajectories of the user's COM to remain inside these regions at all times.

We hypothesize that the size of Stability Basins is correlated with fall risk. A larger Stability Basin implies a larger set of perturbations that can be successfully withstood, and therefore a reduced likelihood that external perturbations would cause failure to complete a task. To test this would require an experiment involving long term observation of a large group of people at risk of falling, and statistically determining whether Stability Basin size indeed predicts fall risk.

In the current method for computing Stability Basins (Sec. 3.5.1), we utilize data from perturbed sit-to-stand trials to help construct a model of the subject's control strategy. We believe that given enough observations of a subject performing sit-to-stand, this requirement would become unnecessary, and an accurate model could be produced from unperturbed data alone. The contributions from Chap. 2 may be useful in this regard, as pose estimation may enable the cheap and accurate collection of kinematic data of subjects performing sit-to-stand outside of laboratory settings.

Cross sections of Stability Basins represent stability during different portions of a given motion [92]. Smaller cross sections therefore indicate instability during a particular time. We may be able to determine which muscles contribute most to a person's instability by identifying the muscles active during these times (i.e. through EMG measurements). We hypothesize that Stability Basins may be useful to therapists working to increase a patient's stability by providing targeted guidance as to which muscles to strengthen.

6.2.3 ARMTD

ARMTD can extend to dynamic obstacles by treating obstacles as static at each time step [181]. This is necessary to operate safely around humans who are moving. The chief challenge here is generating conservative predictions about what the human will do. Additionally, some tasks like helping a person dress may require the manipulator to make contact with a person. ARMTD can still be used in these cases by excluding certain portions of the robot from the collision-avoidance constraints.

Several tweaks to the ARMTD algorithm could reduce its conservatism, improve its plans, and enable its use on even higher degree-of-freedom robots. ARMTD works by computing parameterized reachable sets of sines and cosines of joint angles, which are turned into reachable sets of rotation matrices. This part of the method utilizes zonotopes (which are convex) to overapproximate reachable sets of non-convex arcs of the unit circle. A more complex set representation or a change in reachability algorithm may reduce conservatism at this stage of the pipeline. ARMTD may be extended to include prismatic joints, expanding the range of kinematic chain robots to which the method may be applied.

6.2.4 TRIP-RTD

We are preparing to conduct a perturbative experiment to evaluate TRIP-RTD in practice. Subjects will be tripped by tethers attached to their feet during swing, replicating an experiment performed by previous researchers [3]. After constructing individualized stumble-recovery libraries and target sets for each subject, we will use the same subjects in the second stage of the experiment. Here, subjects will utilize a bypass adaptor with a robotic prosthetic leg and be tripped when the prosthesis is in swing. We hypothesize that the subjects will recover from these perturbations more often when controlling the prosthesis using TRIP-RTD than when nominal prosthetic controller. Future work may also add perception to the TRIP-RTD pipeline [235], and consider user preference when designing the cost function for optimization [236].

6.3 Concluding Remarks

Robots are exciting because they can move; they can interact with the environment, and make decisions about how to move to accomplish their goals. However, robot movement only has a purpose insomuch as it helps people achieve their goals. This can mean performing search and rescue in dangerous situations, transporting packages, or surveying crops. But it can also mean something direct and personal – assisting people with activities of daily living: walking, standing, eating, dressing. When we consider robot motion and autonomy, people living with impaired or reduced mobility can benefit much from these technologies, so long as they prove to be safe and reliable. Much work remains to be done before this future is actualized. Perceiving people, predicting what they will do, and planning in response to their actions are skills that take humans a lifetime to develop, and transferring these skills to robots is a challenging, yet worthwhile endeavor. The tools developed in this dissertation are steps toward a future in which robots safely and autonomously support people in carrying out their daily lives.

BIBLIOGRAPHY

[1] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[2] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

[3] Camila Shirota, Ann M. Simon, and Todd A. Kuiken. Trip recovery strategies following perturbations of variable duration. *Journal of Biomechanics*, 47(11):2679 – 2684, 2014.

[4] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7718–7727, 2019.

[5] Edoardo Remelli, Shangchen Han, Sina Honari, Pascal Fua, and Robert Wang. Lightweight multi-view 3d pose estimation through camera-disentangled representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6040–6049, 2020.

[6] United Nations. World population ageing 2015. pages 1–32, 2015.

[7] Lisa F Berkman, Teresa E Seeman, Marilyn Albert, Dan Blazer, Robert Kahn, Richard Mohs, Caleb Finch, Edward Schneider, Carl Cotman, Gerald McClearn, et al. High, usual and impaired functioning in community-dwelling older men and women: findings from the Macarthur Foundation Research Network on successful aging. *Journal of clinical epidemiology*, 46(10):1129–1140, 1993.

[8] Dilip V Jeste, Gauri N Savla, Wesley K Thompson, Ipsit V Vahia, Danielle K Glorioso, A'verria Sirkin Martin, Barton W Palmer, David Rock, Shahrokh Golshan, Helena C Kraemer, et al. Association between older age and more successful aging: critical role of resilience and depression. *American Journal of Psychiatry*, 170(2):188–196, 2013.

[9] Sandra C Webber, Michelle M Porter, and Verena H Menec. Mobility in older adults: a comprehensive framework. *The gerontologist*, 50(4):443–450, 2010.

[10] Hee Rin Lee and Laurel D Riek. Reframing assistive robots to promote successful aging. *ACM Transactions on Human-Robot Interaction (THRI)*, 7(1):1–23, 2018.

[11] Mirja Hirvensalo, Taina Rantanen, and Eino Heikkinen. Mobility difficulties and physical activity as predictors of mortality and loss of independence in the community-living older population. *Journal of the American Geriatrics Society*, 48(5):493–498, 2000.

[12] Philipp Mahlknecht, Stefan Kiechl, Bastiaan R Bloem, Johann Willeit, Christoph Scherfler, Arno Gasperi, Gregorio Rungger, Werner Poewe, and Klaus Seppi. Prevalence and burden of gait disorders in elderly men and women aged 60–97 years: a population-based study. *PLoS One*, 8(7):e69627, 2013.

[13] M Powell Lawton. Aging and performance of home tasks. *Human factors*, 32(5):527–536, 1990.

[14] Sidney Katz. Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, 31(12):721–727, 1983.

[15] Sumukha Udupa, Vineet R Kamat, and Carol C Menassa. Shared autonomy in assistive mobile robots: a review. *Disability and Rehabilitation: Assistive Technology*, pages 1–22, 2021.

[16] J Kevin Eckert, Leslie A Morgan, and Namratha Swamy. Preferences for receipt of care among community-dwelling adults. *Journal of aging & social policy*, 16(2):49–65, 2004.

[17] Jeff A Small, Gloria Gutman, Saskia Makela, and Beth Hillhouse. Effectiveness of communication strategies used by caregivers of persons with alzheimer's disease during activities of daily living. 2003.

[18] Richard Schulz and Paula R Sherwood. Physical and mental health effects of family caregiving. *Journal of Social Work Education*, 44(sup3):105–113, 2008.

[19] Maja Lopez Hartmann, Johanna De Almeida Mello, Sibyl Anthierens, Anja Declercq, Thérèse Van Durme, Sophie Cès, Véronique Verhoeven, Johan Wens, Jean Macq, and Roy Remmen. Caring for a frail older person: the association between informal caregiver burden and being unsatisfied with support from family and friends. *Age and ageing*, 48(5):658–664, 2019.

[20] Lazaros Penteridis, Grazia D'Onofrio, Daniele Sancarlo, Francesco Giuliani, Francesco Ricciardi, Filippo Cavallo, Antonio Greco, Ilias Trochidis, and Alexander Gkiokas. Robotic and sensor technologies for mobility in older people. *Rejuvenation research*, 20(5):401–410, 2017.

[21] Laurel D Riek. Healthcare robotics. *Communications of the ACM*, 60(11):68–78, 2017.

[22] Martina Čaić, Gaby Odekerken-Schröder, and Dominik Mahr. Service robots: value co-creation and co-destruction in elderly care networks. *Journal of Service Management*, 2018.

[23] Steven W Brose, Douglas J Weber, Ben A Salatin, Garret G Grindle, Hongwu Wang, Juan J Vazquez, and Rory A Cooper. The role of assistive robotics in the lives of persons with disability. *American Journal of Physical Medicine & Rehabilitation*, 89(6):509–521, 2010.

[24] Marion Hersh. Overcoming barriers and increasing independence–service robots for elderly and disabled people. *International Journal of Advanced Robotic Systems*, 12(8):114, 2015.

[25] Laura Fiorini, Marleen De Mul, Isabelle Fabbricotti, Raffaele Limosani, Alessandra Vitanza, Grazia D'Onofrio, Michael Tsui, Daniele Sancarlo, Francesco Giuliani, Antonio Greco, Denis Guiot, Eloïse Senges, and Filippo Cavallo. Assistive robots to improve the independent living of older persons: results from a needs study. *Disability and Rehabilitation: Assistive Technology*, 16(1):92–102, 2021. PMID: 31329000.

[26] Filippo Cavallo, Raffaele Limosani, Laura Fiorini, Raffaele Esposito, Rocco Furferi, Lapo Governi, and Monica Carfagni. Design impact of acceptability and dependability in assisted living robotic applications. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 12(4):1167–1178, 2018.

[27] Xanthi S Papageorgiou, Georgia Chalvatzaki, Costas S Tzafestas, and Petros Maragos. Hidden markov modeling of human normal gait using laser range finder for a mobility assistance robot. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 482–487. IEEE, 2014.

[28] Mineo Ishii, Keijiro Yamamoto, and Kazuhito Hyodo. Stand-alone wearable power assist suit–development and availability–. *Journal of robotics and mechatronics*, 17(5):575–583, 2005.

[29] Evan Ackerman. Toyota research demonstrates ceiling-mounted home robot. *IEEE Spectrum*, Sep 2020.

[30] V Monaco, P Tropea, Federica Aprigliano, Dario Martelli, Andrea Parri, M Cortese, R Molino-Lova, N Vitiello, and Silvestro Micera. An ecologically-controlled exoskeleton can improve balance recovery after slippage. *Scientific reports*, 7:46721, 2017.

[31] Marc G Carmichael and Dikai Liu. Towards using musculoskeletal models for intelligent control of physically assistive robots. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 8162–8165. IEEE, 2011.

[32] Alessandra Vitanza, Grazia D'Onofrio, Francesco Ricciardi, Daniele Sancarlo, Antonio Greco, and Francesco Giuliani. Assistive robots for the elderly: innovative tools to gather health relevant data. In *Data Science for Healthcare*, pages 195–215. Springer, Cham, 2019.

[33] Daehyung Park, Yuuna Hoshi, Harshal P Mahajan, Ho Keun Kim, Zackory Erickson, Wendy A Rogers, and Charles C Kemp. Active robot-assisted feeding with a general-purpose mobile manipulator: Design, evaluation, and lessons learned. *Robotics and Autonomous Systems*, 124:103344, 2020.

[34] Gerard Canal, Guillem Alenya, and Carme Torras. A taxonomy of preferences for physically assistive robots. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 292–297. IEEE, 2017.

[35] Frank Rudzicz, Rosalie Wang, Momotaz Begum, and Alex Mihailidis. Speech interaction with personal assistive robots supporting aging at home for individuals with alzheimer's disease. *ACM Transactions on Accessible Computing (TACCESS)*, 7(2):1–22, 2015.

[36] Grazia D'Onofrio, Daniele Sancarlo, Massimiliano Raciti, Megan Burke, Aimee Teare, Tanja Kovacic, Keith Cortis, Kathy Murphy, Eva Barrett, Sally Whelan, et al. Mario project: validation and evidence of service robots for older people with dementia. *Journal of Alzheimer's Disease*, 68(4):1587–1601, 2019.

[37] Ralf Kittmann, Tim Fröhlich, Johannes Schäfer, Ulrich Reiser, Florian Weißhardt, and Andreas Haug. Let me introduce myself: I am care-o-bot 4, a gentleman robot. In *Mensch und Computer 2015–Tagungsband*, pages 223–232. De Gruyter, 2015.

[38] Laura Fiorini, Raffaele Esposito, Manuele Bonaccorsi, Claudio Petrazzuolo, Filippo Saponara, Roberta Giannantonio, Gianluca De Petris, Paolo Dario, and Filippo Cavallo. Enabling personalised medical support for chronic disease management through a hybrid robot-cloud approach. *Autonomous Robots*, 41(5):1263–1276, 2017.

[39] Ioannis Kostavelis, Dimitrios Giakoumis, Sotiris Malasiotis, and Dimitrios Tzovaras. Ramcip: towards a robotic assistant to support elderly with mild cognitive impairments at home. In *International Symposium on Pervasive Computing Paradigms for Mental Health*, pages 186–195. Springer, 2015.

[40] Tiffany L Chen, Matei Ciocarlie, Steve Cousins, Phillip M Grice, Kelsey Hawkins, Kaijen Hsiao, Charles C Kemp, Chih-Hung King, Daniel A Lazewatsky, Hai Nguyen, et al. Robots for humanity: A case study in assistive mobile manipulation. 2013.

[41] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. Fetch and freight: Standard platforms for service robot applications. In *Workshop on Autonomous Mobile Service Robots*, 2016.

[42] Joy Hammel, Karyl Hall, David Lees, Larry Leifer, Machiel Van der Loos, Inder Perkash, and Robert Crigler. Clinical evaluation of a desktop robotic assistant. *Journal of rehabilitation research and development*, 26(3):1–16, 1989.

[43] HF Machiel Van der Loos, J Joseph Wagner, Niels Smaby, K Chang, Oscar Madrigal, Larry J Leifer, and Oussama Khatib. ProVAR assistive robot system architecture. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 1, pages 741–746. IEEE, 1999.

[44] Won-Kyung Song, He-Young Lee, Jong-Sung Kim, Yong-San Yoon, and Zeungnam Bien. KARES: intelligent rehabilitation robotic system for the disabled and the elderly. In *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Vol. 20 Biomedical Engineering Towards the Year 2000 and Beyond (Cat. No. 98CH36286)*, volume 5, pages 2682–2685. IEEE, 1998.

[45] Zeungnam Bien, Dae-Jin Kim, Myung-Jin Chung, Dong-Soo Kwon, and Pyung-Hun Chang. Development of a wheelchair-based rehabilitation robotic system (KARES II)

with various human-robot interaction interfaces for the disabled. In *Proceedings 2003 IEEE/ASME international conference on advanced intelligent mechatronics (AIM 2003)*, volume 2, pages 902–907. IEEE, 2003.

[46] BJF Driessen, HG Evers, and JA v Woerden. MANUS—a wheelchair-mounted rehabilitation robot. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of engineering in medicine*, 215(3):285–290, 2001.

[47] Veronique Maheu, Philippe S Archambault, Julie Frappier, and François Routhier. Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–5. IEEE, 2011.

[48] Kinova. Testimonial from Laura. `https://www.kinovarobotics.com/en/testimonial-laura`, 2021.

[49] Robert Bogue. Robotic exoskeletons: a review of recent progress. *Industrial Robot: An International Journal*, 2015.

[50] John Brant. At the bionic olympics, engineers and athletes make miracles. *Popular Mechanics*, Apr 2020.

[51] Alberto Esquenazi, Mukul Talaty, Andrew Packel, and Michael Saulino. The ReWalk powered exoskeleton to restore ambulatory function to individuals with thoracic-level motor-complete spinal cord injury. *American journal of physical medicine & rehabilitation*, 91(11):911–921, 2012.

[52] Michael Wehner, Brendan Quinlivan, Patrick M Aubin, Ernesto Martinez-Villalpando, Michael Baumann, Leia Stirling, Kenneth Holt, Robert Wood, and Conor Walsh. A lightweight soft exosuit for gait assistance. In *2013 IEEE international conference on robotics and automation*, pages 3362–3369. IEEE, 2013.

[53] Sikai Zhao, Yeqin Yang, Yang Gao, Zongwei Zhang, Tianjiao Zheng, and Yanhe Zhu. Development of a soft knee exosuit with twisted string actuators for stair climbing assistance. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2541–2546. IEEE, 2019.

[54] Samuel K Au, Jeff Weber, and Hugh Herr. Powered ankle–foot prosthesis improves walking metabolic economy. *IEEE Transactions on robotics*, 25(1):51–66, 2009.

[55] Luke M Mooney, Elliott J Rouse, and Hugh M Herr. Autonomous exoskeleton reduces metabolic cost of human walking during load carriage. *Journal of neuroengineering and rehabilitation*, 11(1):1–11, 2014.

[56] Christopher Kevin Wong, John Rheinstein, and Michelle Stern. Benefits for adults with transfemoral amputations and peripheral artery disease using microprocessor compared with nonmicroprocessor prosthetic knees. *American Journal of Physical Medicine & Rehabilitation*, pages 804–810, 2015.

[57] Alejandro F Azocar, Luke M Mooney, Jean-François Duval, Ann M Simon, Levi J Hargrove, and Elliott J Rouse. Design and clinical implementation of an open-source bionic leg. *Nature biomedical engineering*, 4(10):941–953, 2020.

[58] Michael Tucker, Jeremy Olivier, Anna Pagel, Hannes Bleuler, Mohamed Bouri, Olivier Lambercy, Jose del R. Millan, Robert Riener, Heike Vallery, and Roger Gassert. Control strategies for active lower extremity prosthetics and orthotics: A review. *Journal of NeuroEngineering and Rehabilitation*, 12:1, 01 2015.

[59] David P Miller. Assistive robotics: an overview. *Assistive Technology and Artificial Intelligence*, pages 126–136, 1998.

[60] David Feil-Seifer and Maja J Mataric. Defining socially assistive robotics. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, pages 465–468. IEEE, 2005.

[61] Roger Bemelmans, Gert Jan Gelderblom, Pieter Jonker, and Luc De Witte. Socially assistive robots in elderly care: A systematic review into effects and effectiveness. *Journal of the American Medical Directors Association*, 13(2):114–120, 2012.

[62] Maribel Pino, Mélodie Boulay, François Jouen, and Anne Sophie Rigaud. “are we ready for robots that care for us?” attitudes and opinions of older adults toward socially assistive robots. *Frontiers in aging neuroscience*, 7:141, 2015.

[63] Reza Kachouie, Sima Sedighadeli, Rajiv Khosla, and Mei-Tai Chu. Socially assistive robots in elderly care: a mixed-method systematic literature review. *International Journal of Human-Computer Interaction*, 30(5):369–393, 2014.

[64] Selma Šabanović, Casey C Bennett, Wan-Ling Chang, and Lesa Huber. PARO robot affects diverse interaction modalities in group sensory therapy for older adults with dementia. In *2013 IEEE 13th international conference on rehabilitation robotics (ICORR)*, pages 1–6. IEEE, 2013.

[65] Lillian Hung, Cindy Liu, Evan Woldum, Andy Au-Yeung, Annette Berndt, Christine Wallsworth, Neil Horne, Mario Gregorio, Jim Mann, and Habib Chaudhury. The benefits of and barriers to using a social robot paro in care settings: a scoping review. *BMC geriatrics*, 19(1):1–10, 2019.

[66] Ester Martinez-Martin, Felix Escalona, and Miguel Cazorla. Socially assistive robots for older adults and people with autism: An overview. *Electronics*, 9(2):367, 2020.

[67] Brian Scassellati. How social robots will help us to diagnose, treat, and understand autism. In *Robotics research*, pages 552–563. Springer, 2007.

[68] Paolo Bevilacqua, Marco Frego, Daniele Fontanelli, and Luigi Palopoli. Reactive planning for assistive robots. *IEEE Robotics and Automation Letters*, 3(2):1276–1283, 2018.

[69] Dylan P Losey, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, and Dorsa Sadigh. Controlling assistive robots with learned latent actions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–384. IEEE, 2020.

[70] Przemyslaw A Lasota, Terrence Fong, Julie A Shah, et al. A survey of methods for safe human-robot interaction. *Foundations and Trends® in Robotics*, 5(4):261–349, 2017.

[71] Laurence Rubenstein. Falls in older people: Epidemiology, risk factors and strategies for prevention. *Age and Ageing*, 35 Suppl 2:ii37–ii41, 10 2006.

[72] William C Miller, Mark Speechley, and Barry Deathe. The prevalence and risk factors of falling and fear of falling among lower extremity amputees. *Archives of physical medicine and rehabilitation*, 82(8):1031–1037, 2001.

[73] Maria Vila Abad, Gerard Canal, and Guillem Alenyà. Towards safety in physically assistive robots: eating assistance. 2018.

[74] Eloise Matheson, Riccardo Minto, Emanuele GG Zampieri, Maurizio Faccio, and Giulio Rosati. Human–robot collaboration in manufacturing applications: a review. *Robotics*, 8(4):100, 2019.

[75] Thomas Arnold and Matthias Scheutz. The tactile ethics of soft robotics: Designing wisely for human–robot interaction. *Soft robotics*, 4(2):81–87, 2017.

[76] Sangbae Kim, Cecilia Laschi, and Barry Trimmer. Soft robotics: a bioinspired evolution in robotics. *Trends in biotechnology*, 31(5):287–294, 2013.

[77] Daniel Bruder, Xun Fu, R Brent Gillespie, C David Remy, and Ram Vasudevan. Data-driven control of soft robots using koopman operator theory. *IEEE Transactions on Robotics*, 37(3):948–961, 2020.

[78] Daniel Bruder, Brent Gillespie, C David Remy, and Ram Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. *arXiv preprint arXiv:1902.02827*, 2019.

[79] Daniel Bruder, Xun Fu, and Ram Vasudevan. Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics. *IEEE Robotics and Automation Letters*, 6(3):4369–4376, 2021.

[80] Daniel Bruder, Xun Fu, R Brent Gillespie, C David Remy, and Ram Vasudevan. Koopman-based control of a soft continuum manipulator under variable loading conditions. *arXiv preprint arXiv:2002.01407*, 2020.

[81] J A Stevens, P S Corso, E A Finkelstein, and T R Miller. The costs of fatal and non-fatal falls among older adults. *Injury Prevention*, 12(5):290–295, 2006.

[82] S. Heinrich, K. Rapp, U. Rissmann, C. Becker, and H.-H. König. Cost of falls in old age: a systematic review. *Osteoporosis International*, 21(6):891–902, Jun 2010.

[83] Somil Bansal, Andrea Bajcsy, Ellis Ratner, Anca D Dragan, and Claire J Tomlin. A hamilton-jacobi reachability-based framework for predicting and analyzing human motion for safe planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7149–7155. IEEE, 2020.

[84] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.

[85] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[86] Shreyas Kousik, Bohao Zhang, Pengcheng Zhao, and Ram Vasudevan. Safe, optimal, real-time trajectory planning with a parallel constrained bernstein algorithm. *IEEE Transactions on Robotics*, 37(3):815–830, 2020.

[87] Jean Bernard Lasserre. *Moments, positive polynomials and their applications*, volume 1. World Scientific, 2009. View online.

[88] Pengcheng Zhao, Shankar Mohan, and Ram Vasudevan. Optimal control of polynomial hybrid systems via convex relaxations. *IEEE Transactions on Automatic Control*, 65(5):2062–2077, 2019.

[89] Shreyas Kousik, Sean Vaskov, Fan Bu, Matthew Johnson-Roberson, and Ram Vasudevan. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *The International Journal of Robotics Research*, 39(12):1419–1469, 2020.

[90] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *arXiv preprint arXiv:1601.04037*, 2016. View online.

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

[92] Victor Shia, Talia Yuki Moore, Patrick Holmes, Ruzena Bajcsy, and Ram Vasudevan. Stability basin estimates fall risk from observed kinematics, demonstrated on the sit-to-stand task. *Journal of biomechanics*, 72:37–45, 2018.

[93] Patrick D Holmes, Shannon M Danforth, Xiao-Yu Fu, Talia Y Moore, and Ram Vasudevan. Characterizing the limits of human stability during motion: perturbative experiment validates a model-based approach for the sit-to-stand task. *Royal Society Open Science*, 7(1):191410, 2020.

[94] Patrick Holmes, Shreyas Kousik, Bohao Zhang, Daphna Raz, Corina Barbalata, Matthew Johnson-Roberson, and Ram Vasudevan. Reachable sets for safe, real-time manipulator trajectory design. In *Robotics: Science and Systems*, 2020.

[95] Shreyas Kousik, Patrick Holmes, and Ram Vasudevan. Safe, aggressive quadrotor flight via reachability-based trajectory design. In *Dynamic Systems and Control Conference*, volume 59162, page V003T19A010. American Society of Mechanical Engineers, 2019.

[96] Shannon M Danforth, Patrick D Holmes, and Ram Vasudevan. Trip recovery in lower-limb prostheses using reachable sets of predicted human motion. *arXiv preprint arXiv:2010.11228*, 2020.

[97] Choong Hee Kim, Shannon M Danforth, Patrick D Holmes, Daphna Raz, Darlene Yao, Asheesh Bedi, and Ram Vasudevan. Automated camera-based estimation of rehabilitation criteria following ACL reconstruction. *arXiv preprint arXiv:1810.11087*, 2018.

[98] Patrick Holmes, Shreyas Kousik, Shankar Mohan, and Ram Vasudevan. Convex estimation of the α-confidence reachable set for systems with parametric uncertainty. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 4097–4103. IEEE, 2016. View online.

[99] Joshua G Mangelson, Jinsun Liu, Ryan M Eustice, and Ram Vasudevan. Guaranteed globally optimal planar pose graph and landmark slam via sparse-bounded sums-of-squares programming. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9306–9312. IEEE, 2019.

[100] Matthew Giamou, Filip Maric, Valentin Peretroukhin, and Jonathan Kelly. Sparse bounded degree sum of squares optimization for certifiably globally optimal rotation averaging. *arXiv preprint arXiv:1904.01645*, 2019.

[101] A. J. Campbell, M. J. Borrie, and G. F. Spears. Risk factors for falls in a community-based prospective study of people 70 years and older. *Journal of Gerontology*, 44(4):112–117, 1989.

[102] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[103] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*, pages 291–305. Springer, 2005.

[104] M. Althoff and D. Grebenyuk. Implementation of interval arithmetic in CORA 2016. In *Applied Verification for Continuous and Hybrid Systems*, pages 91–105, 2016.

[105] M. Svenstrup, S. Tranberg, H. J. Andersen, and T. Bak. Pose estimation and adaptive robot behaviour for human-robot interaction. In *2009 IEEE International Conference on Robotics and Automation*, pages 3571–3576, 2009.

[106] Nidhi Seethapathi, Shaofei Wang, Rachit Saluja, Gunnar Blohm, and Konrad P Kording. Movement science needs different pose tracking algorithms. *arXiv preprint arXiv:1907.10226*, 2019.

[107] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

[108] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. Xnect: Real-time multi-person 3d motion capture with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 39(4):82–1, 2020.

[109] Fan Bu, Trinh Le, Xiaoxiao Du, and M. Johnson-Roberson. Pedestrian planar lidar pose (pplp) network for oriented pedestrian detection based on planar lidar and monocular images pedestrian planar lidar pose (pplp) network for oriented pedestrian detection based on planar lidar and monocular images. *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[110] Alex M Andrew. Multiple view geometry in computer vision. *Kybernetes*, 2001.

[111] Chris Aholt, Sameer Agarwal, and Rekha Thomas. A QCQP approach to triangulation. In *European Conference on Computer Vision*, pages 654–667. Springer, 2012.

[112] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.

[113] Haibo Qiu, Chunyu Wang, Jingdong Wang, Naiyan Wang, and Wenjun Zeng. Cross view fusion for 3d human pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4342–4351, 2019.

[114] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Reconstructing 3d human pose from 2d image landmarks. In *European conference on computer vision*, pages 573–586. Springer, 2012.

[115] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7753–7762, 2019.

[116] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[117] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[118] Tillmann Weisser, Jean B Lasserre, and Kim-Chuan Toh. Sparse-bsos: a bounded degree sos hierarchy for large scale polynomial optimization with sparsity. *Mathematical Programming Computation*, 10(1):1–32, 2018.

[119] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 2020.

[120] Eric W. Weisstein. Point-line distance–3-dimensional. *Wolfram MathWorld*, 2021. https://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html.

[121] Kurt M Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43(2-3):471–484, 2009.

[122] Jean B Lasserre, Kim-Chuan Toh, and Shouguang Yang. A bounded degree sos hierarchy for polynomial optimization. *EURO Journal on Computational Optimization*, 5(1-2):87–117, 2017.

[123] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[124] Lucas Brynte, Viktor Larsson, José Pedro Iglesias, Carl Olsson, and Fredrik Kahl. On the tightness of semidefinite relaxations for rotation estimation. *arXiv preprint arXiv:2101.02099*, 2021.

[125] MATLAB Statistics and Machine Learning Toolbox, 2017. The MathWorks, Natick, MA, USA.

[126] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[127] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Godisart, Bart Nabbe, Iain Matthews, et al. Panoptic studio: A massively multiview system for social interaction capture. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):190–204, 2017.

[128] M Clare Robertson, Nancy Devlin, Melinda M Gardner, and A John Campbell. Effectiveness and economic evaluation of a nurse delivered home exercise programme to prevent falls. 1: Randomised controlled trial. *BMJ*, 322(7288):697, 2001.

[129] Bas Bloem, I Boers, M Cramer, R.G.J. Westendorp, and W. Gerschlager. Falls in the elderly. i. identification of risk factors. *Wiener klinische Wochenschrift*, 113:352–62, 2001.

[130] Kim Delbaere, Klaus Hauer, and Stephen Lord. Evaluation of the incidental and planned exercise questionnaire (ipeq) for older people. *British Journal of Sports Medicine*, 44:1029–34, 06 2009.

[131] Klaus Hauer, Lucy Yardley, Nina Beyer, Gertrudis Kempen, N Dias, M Campbell, Clemens Becker, and Chris Todd. Validation of the falls efficacy scale and falls efficacy scale international in geriatric patients with and without cognitive impairment: Results of self-report and interview-based questionnaires. *Journal of Gerontology*, 56:190–9, 10 2009.

[132] Katherine Berg. Measuring balance in the elderly: Preliminary development of an instrument. *Physiotherapy Canada*, 41:304–311, 11 1989.

[133] Diane Podsiadlo and Sandra Richardson. The timed up and go: A test of basic functional mobility for frail elderly persons. *Journal of the American Geriatrics Society*, 39:142–8, 1991.

[134] Lillemor Lundin-Olsson, Lars Nyberg, and Yngve Gustafson. Stops walking when talking as a predictor of falls in the elderly. *Lancet*, 349:617, 04 1997.

[135] Neil D. Weinstein. Unrealistic optimism about susceptibility to health problems: Conclusions from a community-wide sample. *Journal of Behavioral Medicine*, 10:481–500, 11 1987.

[136] Adrian Furnham. Response bias, social desirability and dissimulation. *Personality and Individual Differences*, 7:385–400, 12 1986.

[137] Anne Shumway-Cook, Sandy Brauer, and Marjorie Woollacott. Predicting the probability for falls in community-dwelling older adults using the timed up & go test. *Physical Therapy*, 80(9):896–903, 2000.

[138] Teresa Steffen, Timothy Hacker, and Louise Mollinger. Age- and gender-related test performance in community-dwelling elderly people: Six-minute walk test, berg balance scale, timed up & go test, and gait speeds. *Physical Therapy*, 82:128–37, 02 2002.

[139] Uffe Laessoe, Hans Christian Hoeck, Ole Simonsen, Thomas Sinkjaer, and Michael Voigt. Fall risk in an active elderly population - can it be assessed? *Journal of Negative Results in Biomedicine*, 6:2, 02 2007.

[140] Susan W Muir, Katherine Berg, Bert Chesworth, and Mark Speechley. Use of the berg balance scale for predicting multiple falls in community-dwelling elderly people: A prospective study. *Physical Therapy*, 88(4):449–459, 2008.

[141] Daphna Raz, Edgar Bolivar-Nieto, Necmiye Ozay, and Robert D Gregg. Toward phase-variable control of sit-to-stand motion with a powered knee-ankle prosthesis. In *IEEE Conference on Control Technology and Applications*, 2021.

[142] Jeffrey Hausdorff, Dean A. Rios, and Helen K. Edelberg. Gait variability and fall risk in community-living older adults: A 1-year prospective study. *Archives of Physical Medicine and Rehabilitation*, 82:1050–6, 08 2001.

[143] Thurmon E. Lockhart and Jian Liu. Differentiating fall-prone and healthy adults using local dynamic stability. *Ergonomics*, 51(12):1860–1872, 2008. PMID: 19034782.

[144] Elizabeth Hsiao-Wecksler. Biomechanical and age-related differences in balance recovery using the tether-release method. *Journal of Electromyography and Kinesiology*, 18:179–87, 05 2008.

[145] P. Hur, B. A. Duiser, S. M. Salapaka, and E. T. Hsiao-Wecksler. Measuring robustness of the postural control system to a mild impulsive perturbation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(4):461–467, Aug 2010.

[146] Sjoerd Bruijn, Onno Meijer, Peter Beek, and Jaap Van Dieen. Assessing the stability of human locomotion: a review of current measures. *Journal of the Royal Society Interface*, 11, 01 2014.

[147] Philippe Terrier and Olivier Dériaz. Kinematic variability, fractal dynamics and local dynamic stability of treadmill walking. *Journal of NeuroEngineering and Rehabilitation*, 8(1):12, Feb 2011.

[148] J.B Dingwell, J.P Cusumano, D Sternad, and P.R Cavanagh. Slower speeds in patients with diabetic neuropathy lead to improved local dynamic stability of continuous overground walking. *Journal of Biomechanics*, 33(10):1269 – 1277, 2000.

[149] At Hof, M.G.J. Gazendam, and W.E. Sinke. The condition for dynamic stability. *Journal of Biomechanics*, 38:1–8, 02 2005.

[150] Yi-Chung Pai and James Patton. Center of mass velocity-position predictions for balance control. *Journal of Biomechanics*, 30(4):347 – 354, 1997.

[151] Martin Simoneau and Philippe Corbeil. The effect of time to peak ankle torque on balance stability boundary: experimental validation of a biomechanical model. *Experimental Brain Research*, 165(2):217–228, Aug 2005.

[152] Masahiro Fujimoto and Li-Shan Chou. Dynamic balance control during sit-to-stand movement: An examination with the center of mass acceleration. *Journal of Biomechanics*, 45(3):543 – 548, 2012.

[153] M.A. Hughes, D.K. Weiner, M.L. Schenkman, R.M. Long, and S.A. Studenski. Chair rise strategies in the elderly. *Clinical Biomechanics*, 9(3):187 – 192, 1994.

[154] Rachid Aissaoui and J Dansereau. Biomechanical analysis and modelling of sit to stand task: a literature review. In *IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 141 – 146 vol.1, 02 1999.

[155] LabVIEW. National Instruments.

[156] Visual3D. C-Motion Inc., Germantown, MD, USA.

[157] David A. Winter. *Biomechanics and Motor Control of Human Movement*. John Wiley & Sons, 2005.

[158] Patrick Riley, David E. Krebs, and Rita A. Popat. Biomechanical analysis of failed sit-to-stand. In *IEEE Transactions on Rehabilitation Engineering*, volume 5, pages 353–9, 1998.

[159] Elisabetta Papa and Aurelio Cappozzo. A telescopic inverted-pendulum model of the musculo-skeletal system and its use for the analysis of the sit-to-stand motor task. *Journal of Biomechanics*, 32(11):1205 – 1212, 1999.

[160] AD Kuo. The relative roles of feedforward and feedback in the control of rhythmic movements. *Motor control*, 6:129–145, 04 2002.

[161] Emanuel Todorov. Optimality principles in sensorimotor control. *Nature Neuroscience*, 7(9):907–915, 2004.

[162] D. A. Winter. *The Biomechanics and Motor Control of Human Gait: Normal, Elderly and Pathological*. Waterloo Biomechanics, 01 1991.

[163] Peter Lancaster and L Rodman. *The Algebraic Riccati Equation*, volume 14. 01 1995.

[164] Olaf Stursberg and Bruce H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *Hybrid Systems: Computation and Control*, pages 482–497, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[165] Masahiro Fujimoto and Li-Shan Chou. Region of stability derived by center of mass acceleration better identifies individuals with difficulty in sit-to-stand movement. *Annals of Biomedical Engineering*, 42(4):733–741, Apr 2014.

[166] Yang Wang and Manoj Srinivasan. Stepping in the direction of the fall: the next foot placement can be predicted from current upper body state in steady-state walking. *Biology Letters*, 10(9):20140405, 2014.

[167] Sukyung Park, Fay B. Horak, and Arthur D. Kuo. Postural feedback responses scale with biomechanical constraints in human standing. *Experimental Brain Research*, 154(4):417–427, Feb 2004.

[168] Nils Smit-Anseeuw, C David Remy, and Ram Vasudevan. Walking with confidence: Safety regulation for full order biped models. *IEEE Robotics and Automation Letters*, 4(4):4177–4184, 2019.

[169] Jinsun Liu, Pengcheng Zhao, Zhenyu Gan, Matthew Johnson-Roberson, and Ram Vasudevan. Leveraging the template and anchor framework for safe, online robotic gait design. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10869–10875. IEEE, 2020.

[170] Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Alexis Duburcq, Ayonga Hereid, Omar Harib, Matthieu Masselin, Jessy Grizzle, and Aaron Ames. Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[171] J M Brockway. Derivation of formulae used to calculate energy expenditure in man. *Clinical Nutrition*, 41:463–71, 12 1987.

[172] Jeffrey R Koller, Deanna Gates, Daniel Ferris, and C David Remy. 'body-in-the-loop' optimization of assistive robotic devices: A validation study. In *Robotics: Science and Systems*, 06 2016.

[173] Susanne Oberer and Rolf Dieter Schraft. Robot-dummy crash tests for robot safety assessment. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2934–2939. IEEE, 2007.

[174] F. Pfeiffer and R. Johanni. A concept for manipulator trajectory planning. *IEEE Journal on Robotics and Automation*, 3(2):115–123, 4 1987.

[175] Tobias Kunz and Mike Stilman. Time-Optimal Trajectory Generation for Path Following with Bounded Acceleration and Velocity. In *Robotics: Science and Systems*, 2012.

[176] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001. View online.

[177] Chonhyon Park, Jia Pan, and Dinesh Manocha. Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments. In *Proceedings of the international conference on automated planning and scheduling*, volume 22, 2012.

[178] Sean Murray, Will Floyd-Jones, Ying Qi, Daniel J. Sorin, and George Konidaris. Robot Motion Planning on a Chip. In *Robotics: Science and Systems*, 2016.

[179] T. Kunz, U. Reiser, M. Stilman, and A. Verl. Real-time path planning for a robot arm in changing environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5906–5911, 10 2010.

[180] Kris Hauser. On responsiveness, safety, and completeness in real-time motion planning. *Autonomous Robots*, 32(1):35–48, 1 2012.

[181] Sean Vaskov, Shreyas Kousik, Hannah Larson, Fan Bu, James Ward, Stewart Worrall, Matthew Johnson-Roberson, and Ram Vasudevan. Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments. *Robotics: Science and Systems*, 2019.

[182] Sean Vaskov, Utkarsh Sharma, Shreyas Kousik, Matthew Johnson-Roberson, and Ramanarayan Vasudevan. Guaranteed safe reachability-based trajectory design for a high-fidelity model of an autonomous passenger vehicle, 2019.

[183] Shreyas Kousik, Sean Vaskov, Matthew Johnson-Roberson, and Ram Vasudevan. Safe trajectory synthesis for autonomous driving in unforeseen environments. In *ASME 2017 Dynamic Systems and Control Conference*, pages V001T44A005–V001T44A005. American Society of Mechanical Engineers, 2017. View online.

[184] Sean Vaskov, Hannah Larson, Shreyas Kousik, Matthew Johnson-Roberson, and Ram Vasudevan. Not-at-fault driving in traffic: A reachability-based approach. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2785–2790. IEEE, 2019.

[185] M. Althoff, A. Giusti, S. B. Liu, and A. Pereira. Effortless creation of safe robots from modules through self-programming and self-verification. *Science Robotics*, 4(31), 2019.

[186] A. Singletary, P. Nilsson, T. Gurriet, and A. Ames. Online Active Safety for Robotic Manipulators. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 11 2019.

[187] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin. FaSTrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522, 12 2017.

[188] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin. Decomposition of Reachable Sets and Tubes for a Class of Nonlinear Systems. *IEEE Transactions on Automatic Control*, 63(11):3675–3688, 11 2018.

[189] Shreyas Kousik. *Reachability-based trajectory design*. PhD thesis, 2020.

[190] Thierry Fraichard and James J Kuffner. Guaranteeing motion safety for robots. *Autonomous Robots*, 32(3):173–175, 2012.

[191] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control and verification of high-dimensional systems with dsos and sdsos programming. In *53rd IEEE Conference on Decision and Control*, pages 394–401. IEEE, 2014.

[192] Brad Paden and Ravi Panja. Globally asymptotically stable 'PD+'controller for robot manipulators. *International Journal of Control*, 47(6):1697–1712, 1988.

[193] A. Giusti and M. Althoff. Efficient Computation of Interval-Arithmetic-Based Robust Controllers for Rigid Robots. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 129–135, 4 2017.

[194] Matthias Althoff. *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, Technische Universität München, 2010.

[195] Wikipedia contributors. Rodrigues' rotation formula — Wikipedia, the free encyclopedia, 2021. [Online; accessed 23-May-2021].

[196] Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

[197] Jyh-Ming Lien and Nancy M Amato. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 121–131, 2007.

[198] Leonidas J Guibas, An Nguyen, and Li Zhang. Zonotopes as bounding volumes. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 803–812. Society for Industrial and Applied Mathematics, 2003.

[199] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.

[200] Elijah Polak. *Optimization: algorithms and consistent approximations*, volume 124. Springer Science & Business Media, 2012.

[201] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[202] David Coleman, Ioan Alexandru Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study. *CoRR*, abs/1404.3785, 2014.

[203] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011. View online.

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

[205] Parker Ewen, Jean-Pierre Sleiman, Yuxin Chen, Wei-Chun Lu, Marco Hutter, and Ram Vasudevan. Generating continuous motion and force plans in real-time for legged mobile manipulation. *arXiv preprint arXiv:2104.11685*, 2021.

[206] Kathryn Ziegler-Graham, Ellen J. MacKenzie, Patti L. Ephraim, Thomas G. Travison, and Ron Brookmeyer. Estimating the prevalence of limb loss in the united states: 2005 to 2050. *Archives of Physical Medicine and Rehabilitation*, 89(3):422 – 429, 2008.

[207] Dade D Fletcher, Karen L Andrews, Matthew A Butters, Steven J Jacobsen, Charles M Rowland, and John W Hallett Jr. Rehabilitation of the geriatric vascular amputee patient: a population-based study. *Archives of physical medicine and rehabilitation*, 82(6):776–779, 2001.

[208] William C. Miller, Mark Speechley, and Barry Deathe. The prevalence and risk factors of falling and fear of falling among lower extremity amputees. *Archives of Physical Medicine and Rehabilitation*, 82(8):1031 – 1037, 2001.

[209] A. J. Blake, K. Morgan, M. J. Bendall, H. Dalloso, S. B. J. Ebrahim, T. H. D. Arie, P. H. Fentem, and E. J. Bassey. Falls by elderly people at home: prevalence and associated factors. *Age and Ageing*, 17(6):365–372, 11 1988.

[210] F. Sup, A. Bohara, and M. Goldfarb. Design and control of a powered knee and ankle prosthesis. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4134–4139, 2007.

[211] H. Huang, T. A. Kuiken, and R. D. Lipschutz. A strategy for identifying locomotion modes using surface electromyography. *IEEE Transactions on Biomedical Engineering*, 56(1):65–73, 2009.

[212] Daniel P. Ferris, Keith E. Gordon, Gregory S. Sawicki, and Ammanath Peethambaran. An improved powered ankle–foot orthosis using proportional myoelectric control. *Gait & Posture*, 23(4):425 – 428, 2006.

[213] S. Rezazadeh, D. Quintero, N. Divekar, E. Reznick, L. Gray, and R. D. Gregg. A phase variable approach for improved rhythmic and non-rhythmic control of a powered knee-ankle prosthesis. *IEEE Access*, 7:109840–109855, 2019.

[214] A.M. Schillings, B.M.H. Van Wezel, TH. Mulder, and J. Duysens. Widespread short-latency stretch reflexes and their modulation during stumbling over obstacles. *Brain Research*, 816(2):480 – 486, 1999.

[215] B. E. Lawson, H. Atakan Varol, F. Sup, and M. Goldfarb. Stumble detection and classification for an intelligent transfemoral prosthesis. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 511–514, 2010.

[216] F. Zhang, S. E. D'Andrea, M. J. Nunnery, S. M. Kay, and H. Huang. Towards design of a stumble detection system for artificial legs. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(5):567–577, 2011.

[217] R. D. Gregg, T. Lenzi, N. P. Fey, L. J. Hargrove, and J. W. Sensinger. Experimental effective shape control of a powered transfemoral prosthesis. In *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, pages 1–7, 2013.

[218] S. K. Au, J. Weber, and H. Herr. Powered ankle–foot prosthesis improves walking metabolic economy. *IEEE Transactions on Robotics*, 25(1):51–66, 2009.

[219] Nitish Thatte, Nandagopal Srinivasan, and Hartmut Geyer. Real-time reactive trip avoidance for powered transfemoral prostheses. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, 2019.

[220] Camila Shirota, Ann M. Simon, and Todd A. Kuiken. Transfemoral amputee recovery strategies following trips to their sound and prosthesis sides throughout swing phase. *Journal of Neuroengineering and Rehabilitation*, 12, 2015. Copyright - Copyright BioMed Central 2015; Last updated - 2016-08-07.

[221] Maura E. Eveld, Shane T. King, Leo G. Vailati, Karl E. Zelik, and Michael Goldfarb. On the Basis for Stumble Recovery Strategy Selection in Healthy Adults. *Journal of Biomechanical Engineering*, 143(7), 04 2021. 071003.

[222] 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. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 1, pages 239–246 vol.1, 2001.

[223] R. D. Gregg and J. W. Sensinger. Towards biomimetic virtual constraint control of a powered prosthetic leg. *IEEE Transactions on Control Systems Technology*, 22(1):246–254, 2014.

[224] Varun Joshi and Manoj Srinivasan. A controller for walking derived from how humans recover from perturbations. *Journal of The Royal Society Interface*, 16(157):20190027, 2019.

[225] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, 2013. http://control.ee.ethz.ch/~mpt.

[226] A. F. Azocar, L. M. Mooney, L. J. Hargrove, and E. J. Rouse. Design and characterization of an open-source robotic leg prosthesis. In *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, pages 111–118, 2018.

[227] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[228] Yifei Simon Shao, Chao Chen, Shreyas Kousik, and Ram Vasudevan. Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control, 2021.

[229] Pedro Hespanhol, Matthew Porter, Ram Vasudevan, and Anil Aswani. Dynamic watermarking for general lti systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1834–1839. IEEE, 2017.

[230] Matthew Porter, Arnav Joshi, Pedro Hespanho, Anil Aswani, Matthew Johnson-Roberson, and Ram Vasudevan. Simulation and real-world evaluation of attack detection schemes. In *2019 American Control Conference (ACC)*, pages 551–558. IEEE, 2019.

[231] Matthew Porter, Pedro Hespanhol, Anil Aswani, Matthew Johnson-Roberson, and Ramanarayan Vasudevan. Detecting generalized replay attacks via time-varying dynamic watermarking. *IEEE Transactions on Automatic Control*, 2020.

[232] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[233] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation, 2020.

[234] Taosha Fan, Kalyan Vasudev Alwala, Donglai Xiang, Weipeng Xu, Todd Murphey, and Mustafa Mukadam. Revitalizing optimization for 3d human pose and shape estimation: A sparse constrained formulation, 2021.

[235] Ung Hee Lee, Justin Bi, Rishi Patel, David Fouhey, and Elliott Rouse. Image transformation and cnns: A strategy for encoding human locomotor intent for autonomous wearable robots. *IEEE Robotics and Automation Letters*, 5(4):5440–5447, 2020.

[236] Kimberly A Ingraham, C David Remy, and Elliott J Rouse. User preference of applied torque characteristics for bilateral powered ankle exoskeletons. In *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, pages 839–845. IEEE.