Navigation and Physical Interaction with Balancing Robots

Michael Shomin

CMU-RI-TR-16-58

October 20, 2016

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Ralph Hollis, Chair
Jodi Forlizzi
George Kantor
Bill Smart, Oregon State University

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Abstract

This work describes methods for advancing the state of the art in mobile robot navigation and physical Human-Robot Interaction (pHRI). An enabling technology in this effort is the ballbot, a person-sized mobile robot that balances on a ball. This underactuated robot presents unique challenges in planning, navigation, and control; however, it also has significant advantages over conventional mobile robots. The ballbot is omnidirectional and physically compliant. Moving requires the ballbot to lean, but this also gives it the ability to achieve both soft, compliant physical interaction and apply large forces.

The work presented in this dissertation demonstrates the ability to navigate cluttered environments with the ballbot. Formulating the system as differentially flat enables fast, analytic trajectory planning. These trajectories are used to plan in the space of static and dynamic obstacles. Leveraging the ballbot's navigational capabilities, this dissertation also presents a method of physically leading people by the hand. A human subject trial was conducted to assess the feasibility, safety, and comfort of this method. This study was successful, with the ballbot leading participants to multiple goals utilizing an amount of force that users found comfortable.

Another area of pHRI explored in this dissertation is assisting people in transition from a seated position to standing. Another user study was conducted to discover how humans help each other out of chairs and how much force they apply. These data were used to design an impedance controller for the ballbot, and this controller was tested and found to deliver equivalent forces to those generated by people.

Lastly, this work explores capabilities that could enable the ballbot to navigate through dense crowds of people. A method for detecting collision and estimating external forces was explored. This method was tested and used to modify a costmap. Iteratively updating this costmap and using it to plan trajectories enabled the robot to discover obstacles through collision. Because the ballbot is inherently compliant, these collisions resulted in safe interactions with small forces.

Acknowledgments

The work and effort required for a PhD is nearly impossible to achieve alone. My thesis is no exception, and I would like to thank those who have helped me in my journey. I would first like to thank all the people who make the Robotics Institute what it is. This place is truly special, and I am so thankful to have been a part of it.

My labmates (MSLians) have been so critical to my work, helping me run experiments and discuss theories and methods. I would like thank Umashankar Nagarajan for all of his assistance in getting me started on the project and all of his work on the ballbot. My other labmates have helped run countless tests with ballbot and collaborated on so many ideas. Thank you Ankit Bhatia, Bhaskar Vaidya, Greg Seyfarth, Olaf-Dietrich Sanick, Chanapol Puapattanakun, and Tekin Mericli. I would also like to thank John Antanitis for his help conducting human subject trials. During my time in the lab, I've had the pleasure of working with and mentoring many summer students. This has been a fantastic teaching experience, but I have also learned so much from them. I would like to thank these students: Kenneth Payson, Carl Curren, Yufan (Adam) Zhang, Mark Dudley, Anson Wang, and Mike Lee. Although I did not collaborate with all the members of MSL, I did get to know them all. I am so fortunate to have had technical discussions, ask for advice, and become friends with these great people. Thank you Masaaki Kumagai, Camino Fernandez Llamas, Garth Zeglin, John Kozar, Teppei Tsujita, and Sebastian Schoendorfer.

Outside of the lab, I have made so many fantastic friends in the Robotics Institute. There are far too many to mention, but please know that I am so thankful to have such great and supportive people in my life.

My committee member, George Kantor, has given me advice and direction since I started working on the ballbot. I appreciate our many conversations about ideas and results. Thank you George. I would also like to thank Jodi Forlizzi, who has taught me a tremendous amount about creating positive interactions with robots and conducting subject trials. Thank you, also to my external committee member, Bill Smart. Our discussions about HRI and social navigation have been hugely helpful to me and my work.

The person who made this work possible, my advisor Ralph Hollis deserves my deepest gratitude. It has been a pleasure to work with you, and I have learned so much from your mentorship. The way you approach robotics, fabrication, and design as well as your work ethic has influenced me more than you know. I would also like to thank Beth Hollis for brownies, banana bread, and generally being an amazing person. Thank you both for your incredible support and kindness.

Lastly, I would like to thank my family. It has been a long road, and I wouldn't have made it here with your support. I especially need to thank my wife, Liz. Being PhD students and living in different cities has certainly not been easy, but I couldn't have done it without you. Thank you for the years of support and motivation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The fields of manipulation, human-robot interaction, navigation, and many others have brought robotic technology to a stage of real value to people outside the field. There are, however, still a myriad of opportunities for improvement, especially in the field of physical interaction with humans.

Physical Human-Robot Interaction (pHRI) is a blossoming field within robotics, with the potential to change the way that average people perceive and work with robots. Until now, this field has been mostly restricted to manipulators on a table. Such systems are important to solving basic research questions of safety of operation around people but incredibly restrictive in the types of interaction achievable. By extending the concepts of physical interaction to dynamic mobile robots, many new possibilities arise.

In this dissertation, we aim to explore the capabilities of a balancing mobile robot that makes contact with the external world in meaningful ways. The mobile robot of interest here is the ballbot [Lauwers et al., 2005], shown in Fig. 1.1(a), a person sized robot that balances on a ball. The ballbot was invented by Ralph Hollis and developed by his research group [Hollis, 2010] (PAT US7847504 B2). It is a member of a group of dynamically balancing mobile robots that present significant challenges in control, planning, and navigation. These challenges, however, are accompanied by many advantages over traditional robotic platforms that make the ballbot uniquely suited to physical interaction with people. Physical interaction becomes important in a variety of scenarios, including force-based assistance, and navigation of dense crowds.

1.1 Motivating Domains

This work focuses on balancing mobile robots, utlizing the ballbot as an experimental system. Broadly, the concepts presented in this thesis are generalizable to multiple domains within robotics, including navigation and physical human-robot interaction (pHRI).

1.1.1 Navigation among Clutter

Although robot navigation has been studied for many years, it remains an unsolved problem in some domains. This is especially true of large robots operating in unstructured, indoor environ-

ments. Deploying robots that help people can be made significantly easier if those robots are capable of navigating spaces that were initially set up for humans, not robots.

(a) The ballbot navigating through a doorway. This maneuver is made easier because the robot has a relatively small footprint for its size.

(b) A demonstration of the ballbot's physical compliance. A small child is pushing the 65 kg robot, and it moves away.

Figure 1.1: The ballbot, a person-sized, balancing mobile robot.

The ballbot is uniquely suited to navigate through cluttered environments intended for people because of its advantageous aspect ratio, inherent compliance, and smooth graceful motion. The ballbot can easily move through a standard doorway with room to spare as seen in Fig. 1.1(a). This is because the ballbot is the width and height of a person. The details of navigation will be discussed in Chapter 5.

1.1.2 Navigation through Crowds

One recent area of study has been the SPENCER project, an assistive robot operating in a busy airport [Triebel et al., 2015]. Other work has investigated a robot navigating through city streets [Lidoris et al., 2009]. Historically, Nourbakhsh et al implemented a functional tour guide robot with collision avoidance at the Carnegie Museum of Natural History [Nourbakhsh et al., 1999]. These studies demonstrate that robots can move through public human spaces. However, they also show key limitations in crowd density and speed because all methods that we are aware of completely exclude planning and use of trajectories that touch people. Generally, this is a good thing, as robots are not expected to make contact with people. We posit, however, collision is not only inevitable, but also helpful in some cases.

Fig. 1.2 shows an airport terminal being evacuated with nearly every person in this scene touching at least one other person. For a robot to navigate through this group, contact would be imminent and necessary. In such a scenario, a robot could be very useful, as it could navigate

Figure 1.2: A very crowded airport would present a very difficult navigation environment for a robot. Of note, many people contact one another in this setting.

through an evacuated area, searching for danger without putting any humans in harm's way. A balancing robot that can intentionally make contact with people to navigate through dense crowds would be well-suited to handle such an event.

1.1.3 Physical Assistance to Humans

In 2011, hospital workers' injury rate from overexertion was twice that of the average of all industries. Nursing home workers were more than three times more likely to be injured with moving and lifting of patients as the greatest risk factor [BLS, 2011]. This is an area where robots can and should help. Sec. 6.2 describes prior work in helping people out of chairs using the ballbot, operated as a large "force-controlled" robot by using its lean angle to apply forces to people. This is a unique ability of the ballbot's, as it can apply large forces, up to 120 N [Shomin et al., 2015], yet is still safe and compliant around people.

Fig. 1.1(b) shows a child pushing the ballbot, inducing a velocity making it move away. Inherent compliance makes the ballbot well suited to such physical interaction. Despite the robot weighing over 65 kg, a child can exert merely a 3 N force to move the robot away.

Combining this compliant, force-based interaction with navigation capabilities opens new doors in assistive technology. We hypothesize that this type of interaction has potential in health care, rehabilitation, elder care, and assistance to visually impaired people. Fig. 1.3 shows a child being helped to cross an intersection by an adult (a) and a blind man being led by the hand (b). If physical assistance such as this could be achieved with a robot, new mobility and care opportunities could be available in the future. Robotic navigational assistance could be an enabling technology to elderly people as well [Heerink et al., 2006], allowing assisted mobility without the need for additional injury to healthcare workers.

© Amanda Mills, "Paused at an intersection" April 2, 2015 via public-domain-image.com

(a) A father holds his daughter's hand while waiting to cross the street.

© Emilio Labrador, "Assisting blind man walking Mexico" October 16, 2012 via Wikimedia

(b) A blind man is assisted through physical interaction.

Figure 1.3: Two examples of physical human-human interaction.

1.2 Thesis Objectives

We propose using a balancing robot to forcefully interact with the world through lean angle and position control. By designing controllers that yield desired open loop force output, but close the loop using attitude and localization information, robust interaction can be achieved with simplified models. Namely, we propose three useful applications of such a technique:

Forceful Interaction with People

Leveraging the compliance and dynamics of the ballbot, we aim to interact with people using the robot's arms. The ballbot will demonstrate competent physical interaction with both large and small forces. Specifically, the ballbot will apply large forces to assist persons out of a chair, from sit-to-stand. Small forces will be used to lead a person by the hand, through a known indoor environment.

Balancing with Planned and Unexpected Collisions

Using insight gained in prior work [Shomin et al., 2015], a dynamic model of the ballbot along with real time sensor feedback will allow force estimation of objects in collision with ballbot. Such a capability would permit the ballbot to recognize when it is in contact with objects, infer the forces acting, and update models as necessary. If successful, this is an enabling technology for navigation through crowds with contact.

Planning and Navigation in Dynamic Environments with Compliance

Navigation through cluttered environments both with and without humans will be a necessity for assistive technologies. We will achieve this goal in the presence of external forces, both from obstacles in the world and people holding the arm or hand of the ballbot. This feature in particular has very interesting applications to increasing mobility of the visually impaired and elderly.

1.3 Approach

To achieve the goals of this thesis, a variety of techniques in planning, control, and estimation are necessary:

Simplification of Planning via Differential Flatness

Using a differentially flat parameterization of the ballbot model allows for real time planning and control, permitting fast navigation in the face of a changing environment.

Fast, Dynamic Replanning

Using a Model Predictive Control-like approach to trajectory following, control of the system can be made substantially more robust, facilitating tight attitude control.

Forces through Attitude Control

Robust linear control laws on attitude can result in bounded and useful output in the force domain.

Force Estimation for Costmap Inference

By estimating the externally acting forces on the robot during navigation, a more useful cost-to-go can be inferred for use in planning.

1.4 Thesis Outline

The rest of this document is organized as follows. Chapter 2 contains an overview of published work in the fields relavent to this thesis. Chapter 3 introduces the ballbot. The history of the platform as well as the dynamic model is presented. This chapter also describes the hardware and software architecture of the system. Chapter 4 discusses the formulation of the ballbot as differentially flat, an enabling step in real time planning. This section shows the mathematical formulation, assumptions, and method of generating feasible trajectories for the ballbot. The method presented is analytic and very computationally efficient, which facilitates its use for navigation. Chapter 5 highlights how the properties of flatness are used for planning and navigation, extending feasible trajectory generation to planning in the space of obstacles. Relying on the analytic formulation, this method is also very efficient allowing millisecond planning times. Chapter 6 explores physical interaction with the ballbot. It is comprised of three main sections. Section 6.2 presents work on assisting people from a seated position to standing. This line of research involved a user study measuring forces and joint tracking of a human-human interaction study. The realization of a functional controller on the ballbot was not only an important step toward more physical interaction, but provides a key insight into applying forces with the ballbot. Section 6.3 describes a method of physically leading people by the hand. This section also presents the results of a human subject trial that tested and validated the method. Lastly, Section 6.4 introduces work toward navigation through dense human crowds. This included detection and estimation of collisions and using said collisions to plan better trajectories. Finally, Chapter 7 contains concluding remarks, the contributions of this thesis, and potential future work.

Chapter 2

Related Work

Navigation while in contact with people and objects presents scientific problems necessarily spanning multiple fields of robotics. As such, there is vast related work in human-robot interaction (HRI), planning, modeling, and simulation. This chapter gives an overview of relevant material to this thesis.

Section 2.1 presents an overview of relevant literature in physical Human-Robot Interaction. Our research falls squarely in the field, although it is unique in many of the approaches. Most of the research in the field is focused on serial link manipulators, not mobile robots; however, many concepts are very applicable.

Section 2.2 discusses the concept of Differential Flatness. This mathematical concept allows for vastly reduced computational load to plan for some dynamic systems. Our experimental platform is introduced as a differentially flat system in Section 4.2; however, the methods employed rely heavily on prior work in the field discussed in this section.

Section 2.3 presents many approaches to navigating amongst people. Modeling how people move in the presence of robots is well-studied and discussed in this section. This section also highlights prior research in simulating pedestrians for the purposes of navigation as well as graphics. Lastly, different planning approaches for robots avoiding collision with people will be discussed.

2.1 Physical Human-Robot Interaction

Physical HRI entails the research of safe robot operation around humans. Most research historically utilized fixed-base robotic arms, but the field also includes research on human-robot collaboration and proxemics among many other subjects. For the goals of this work, we are interested primarily in collision, force detection, and robotic assistance.

Robots operating around people need to be safe, and this is sometimes achieved through weak robots that cannot physically harm people. Clearly this comes at the cost of capability, so many researchers have investigated online collision and force detection. Various methods have been explored, but a very successful approach uses a residual function [Luca et al., 2006] to evaluate how well an arm is tracking its desired behavior. This scalar function can simply be thresholded to determine collision state. Such an approach is encouraging in its simplicity and robustness,

and it will be further discussed in Section 6.4.2. The same research group extended this work [Haddadin and Albu-Schaffer, 2008] to the estimation of contact force, and quick reactions to contact, reducing contact force. Their experiments tested robot arm speeds up to 2.7 m/s. Further development led to incorporating a point cloud sensor [Magrini et al., 2015] to localize the point on the arm where the contact occurred. These studies established a method to detect and localize collisions.

One area of research very similar to our goal of navigation in dense crowds is contact-based inducement [Shrestha et al., 2015]. This project used a mobile robot with an arm for navigation, where the robot nudged people with the arm in an attempt to make them move. Their study was the first performed where a robot physically interacted (on purpose) with human subjects during active navigation. These interactions were treated as a motion primitives in an Rapidly-expanding Random Tree (RRT) planning framework.

Kulyukin et al. used RFID tags to localize a robot in a building. In addition, they performed a user study where their robot led visually impaired people tethered to their robotic platform [Kulyukin et al., 2004]. The forceful interaction was neglected, but their results provide support that robots can provide assistive capability through a building. Assistive navigation also provides opportunities to positively help elderly people in their activities of daily living. Though not explicitly studying physical interaction, Heerink et al. found that social robots could make elderly people feel comfortable [Heerink et al., 2006] and that interaction with a robot invited expressiveness from elders in a care facility. Overall these studies provided support that robots may successfully be implemented to provide physical assistance in a comfortable way.

2.2 Differential Flatness

Navigation with a balancing robot was a major goal of this thesis work, and our approach relies heavily on fast, dynamic planning enabled by the concept of differential flatness. This method will be discussed in more detail as it relates to the ballbot in Section 4.2.

Differential Flatness [Fliess et al., 1995] is essentially an exact model reduction that simplifies planning for a system by reducing dimensionality enabling the planning of analytic, exact trajectories. This is often faster than sampling-based methods and does not need to be discretized. The computational saving is similar to the use of motion primitives in sampling-based and graph search approaches, but has much more flexibility.

Using differential flatness for real-time trajectory generation was first explored with application to a vectored-thrust flight system [Van Nieuwstadt and Murray, 1998]. Trajectory generation via flatness was then extended to use inequality constraints [Faiz et al., 2001]. The principles shown by these studies are useful in planning amongst obstacles.

Many of the algorithms and much of the intuition for navigation of balancing robots comes from prior work in trajectory generation and planning for quadrotors. Flatness was first used to generate trajectories of single polynomials as well as multi-waypoint paths using the concept of minimum snap [Mellinger and Kumar, 2011]. Because quadrotors behave as a third order system, minimizing snap is analogous to minimum jerk trajectories in second order systems. This motivates the minimization of crackle, the time derivative of snap, in the ballbot.

Most trajectory generation that relies on flatness uses polynomial basis functions for the plan-

ning variables (flat outputs). Flores et al. explored the use of different basis functions [Flores and Milam, 2006], specifically non-uniform rational b-splines. This formulation made the numerical computation much more efficient for planning multi-segment trajectories than optimization over polynomial coefficients. Further study into basis function was also explored in the area of quadrotors by explicitly capturing dynamic contraints and consistency contraints into the basis functions [Richter et al., 2013]. Reformulating the problem this way created a framework for planning multi-segment trajectories with no constraints, enabling very significant computational advantages. This concept is used and extended to plan for balancing mobile robots in Chapter 5

2.3 Navigation among Humans

Robots in many domains benefit from the ability to navigate through cluttered environments; however, these environments may be further congested by people. Planning in a space that contains people is uniquely difficult, as people are dynamic, and sometimes unpredictable.

Many researchers have investigated mobile robot navigation amongst people, using a variety of techniques. These approaches vary based on context, density of people, type of robot, and speed. The relevant research areas can broadly be separated in modeling people and planning. Also applicable to this domain is the simulation of people, both for robotic navigation and graphics.

2.3.1 Simulation

Simulating realistic pedestrian movement is important for video games and cinema and has been studied using a variety of approaches. Shao et al. developed a system for simulation of large groups of pedestrians each with individual perception and planning in large environments for use in animation [Shao and Terzopoulos, 2007]. A similar technique was proposed and extended to include predictive collision avoidance for use in video games [Karamouzas et al., 2009]. This simple collision avoidance model exhibited some interesting emergent behavior such as lane forming.

Outside of graphics and games, some researchers have sought to create models of walking people that can be experimentally verified. Pettre et al. posited a pairwise social force model that they then tuned and verified with a user study[Pettré et al., 2009]. Two participants were sent on colliding trajectories and avoidance strategies were observed and verified to agree with the model. Paris et al. also worked to verify a model using motion capture data [Paris et al., 2007]. They found that long-term prediction can be necessary in some situations and that model parameters should adapt to environmental context.

Some open-source software is especially helpful in the simulation of crowds and could be particularly helpful in the proposed research of this thesis. Pedsim [Cavens et al., 2007] is a simulator that makes distributed intelligence using discrete models easy to implement and interact with, and integration with ROS [Quigley et al., 2009], the Robot Operating System, has already been done. The Pedsim ROS module uses a model of social force that utilizes Langevin equations [Helbing and Molnár, 1995] and shows very realistic simulation of large crowds.

2.3.2 Modeling and Tracking

SPENCER is a multi-university EU research project [Triebel et al., 2015] aimed at the creation of a socially compliant assistive robot for use in airports. The project is very multi-faceted, incorporating planning, sensing, localization, and learning models of people. Specifically, the project learns models of social behavior of person-person and person-robot interactions and uses these models for planning paths.

Related is work from the University of Freiburg [Linder and Arras, 2014] to track multiple social groups using RGB-D sensors. Experiments were conducted using an outdoor robot demonstrating real time tracking of groups of pedestrians. One key motivation comes from a finding that 70% of pedestrians travel in groups.

Achieving better tracking of people has also been studied with explicit incorporation of a social force model [Luber et al., 2010]. Tracking people has been attempted with a variety of sensor modalities, but for robust real time performance, scanning laser range finders are a popular choice, as in the open-source software tool, People2D [Spinello and Siegwart, 2008].

Recognition of a person's intent can give a mobile robot useful information for planning. Hung et al. used worn accelerometers to detect conversing groups [Hung et al., 2014]. Where someone is moving to is probably the most useful aspect of intent for planning, as investigated by Thompson et al. This probabilistic framework reasoned about expected goal locations of moving people for collision avoidance using a laser scanner [Thompson et al., 2009].

2.3.3 Planning

Robot motion planning amongst people has been studied by a number of research groups using different techniques. Vasquez et al. investigated Inverse Reinforcement Learning [Vasquez et al., 2014]. Their method was evaluated on simulations of queues, a crossing hallway, and a four-way intersection. They found that the features for learning were far more important than the learning algorithm. Additionally, collision avoidance required motion prediction.

Spatial behavior was also investigated by Chung et al. using a model of person-person and person-environment interactions [Chung and Huang, 2010]. They found that such a model generated socially acceptable paths. Wolfram Burgard's group at University of Freiburg took a slightly different approach to the problem by teaching socially acceptable movements through teleoperation [Kuderer and Kretzschmar, 2013]. Using feature-based maximum entropy learning, they generated a policy that estimates a probability distribution over trajectories for cooperative avoidance of collisions. The additional consideration of uncertainty in future collision was included in the work of Althoff et al. [Althoff et al., 2011]. This study estimated the future collision probability under uncertainty and chose the path with minimum risk.

Two research groups undertook very large data collection and experiments while investigating navigation among people. The Autonomous City Explorer (ACE) project at the Technical University of Munich [Lidoris et al., 2009] deployed a robot capable of SLAM and navigation in the streets of Munich. They found it to be capable of safe, long-term autonomy. Trautman et al. investigated indoor crowds with higher densities of people (.8 humans per square meter) and developed a model of person-robot cooperative collision avoidance using 488 robot trials [Trautman et al., 2015].

Layered costmaps have also been utilized in planning amongst people [Lu et al., 2014]. Work by Lu et al. presented software and strategies enabling costmap-based planners to use different costmap layers with various properties for navigation. Social navigation is an especially compelling use of layered costmaps, as obstacle inflation can be tuned specifically to people independently of obstacles and the static maps. Implementation in ROS [Quigley et al., 2009] also makes this software especially helpful for experimentation. Lu et al. have also investigated tuning costmap parameters for social navigation[Lu et al., 2013].

Chapter 3

Ballbot

This thesis focuses on algorithms and experiments using the ballbot. This robot offers unique advantages over other mobile robots and also presents unique challenges in planning and control. This chapter introduces the ballbot and the context for the subsequent chapters starting with a short background on the history of ballbot. This is followed by descriptions of the physical system and the system's dynamic model used for analysis and planning. The hardware and software necessary for operation and experimentation is described next. Finally, the experimental setup for testing the algorithms of the work in this thesis is introduced.

3.1 Background

The ballbot is an omnidirectional, dynamically stable mobile robot. It is a human-sized robot that balances on a single spherical wheel. As an underactuated, shape-accelerated system, it moves by leaning and cannot directly control its position. Similar to the classical controls problem of a cart-pole inverted pendulum, but with a few key differences discussed later, the ballbot presents many interesting dynamics and controls problems. Ballbot provides many advantages over a statically stable robot [Lauwers et al., 2005] including having inherent compliance and small footprint. Being tall and thin allows easier navigation amongst cluttered human environments.

The ballbot was invented by Ralph Hollis in 2005 at Carnegie Mellon University [Hollis, 2006]. It was originally designed to be of a human form factor, both in footprint and height. Early capabilities of balancing and station-keeping were demonstrated by Tom Lauwers [Lauwers et al., 2005, 2006]. Anish Mampetta further developed the robot to transition to static stability using its three legs in 2006 [Mampetta, 2006].

Critical to the work of this thesis, Eric Shearer explored the implications of arms to the ballbots dynamics by performing simulation [Schearer, 2006]. Byunjun Kim furthered this work by designing and implementing arms on the ballbot in 2011.

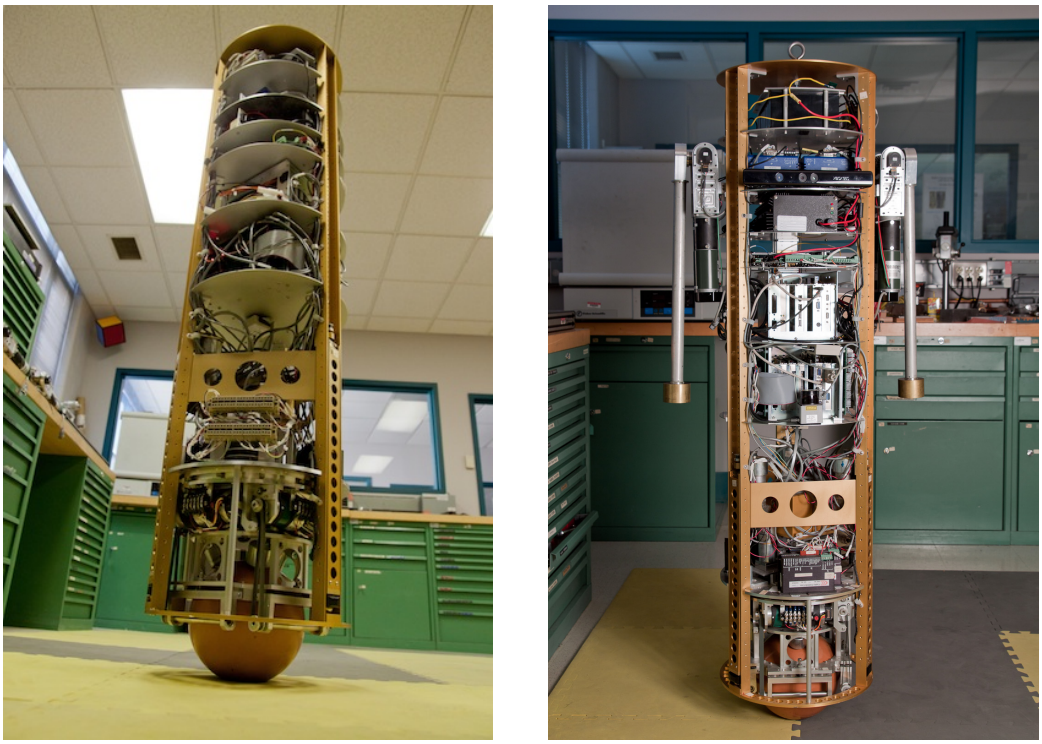
Umashankar Nagarajan joined the project in 2007 and contributed significantly to the ballbot's capabilities over the next five years. In 2009, he demonstrated a strategy for transitioning between balancing and station-keeping based on the magnitude of exerted forces from a person [Nagarajan et al., 2009a]. This seminal work provided the basis for the ballbot's use as a platform for physical Human-Robot Interaciton (pHRI). Also in 2009, the group developed a

strategy for generating parametric lean angle trajectories based on hyperbolic trigonometry functions [Nagarajan et al., 2009b]. This work was further refined into an optimization algorithm for generating optimal state trajectories with specified boundary conditions [Nagarajan, 2010] in 2010. These trajectories were dynamically feasible for the ballbot, a nontrivial task.

In 2011 and 2012, Umashankar Nagarajan extended his work in trajectory optimization to facilitate navigation amongst obstacles. He developed "Motion Policies" [Nagarajan et al., 2012a], optimized motion primitives with an associated controller for a given domain. By testing the regions of attraction in simulation for these optimized trajectories, these motion primitives could be chained together to form longer trajectories. These trajectories could be used to span a known space and then planned using graph search [Nagarajan, 2012; Nagarajan and Hollis, 2013]. A comprehensive description of the ballbot's capabilities using this strategy was presented in 2014 [Nagarajan et al., 2014].

3.2 Ballbot Description

The ballbot is a 1.5 m tall cylindrical robot that balances on top of a 185 mm ball. The robot can be seen with and without its arms in Fig. 3.2.



(a) Ballbot balancing on its single spherical wheel

(b) Ballbot with arms attached held to the side

Figure 3.1: Ballbot, an omnidirectional mobile robot that balances on a ball

The ballbot moves the ball wheel on which it balances through its "Inverse Mouse Ball Drive"

(IMB) [Lauwers et al., 2005]. This mechanism squeezes the ball with four rollers connected to motors via pulleys and belts, as shown in Fig. 3.1(a). Omnidirectional motion is enabled by moving ball in any direction using this mechanism. This means that the ballbot does not have to turn to face its direction of motion. Turning the body is, however, sometimes necessary to orient sensors on the body to a task or to position its arms. As such, the robot can also yaw relative to the IMB via another motor, bearing, and slip ring.

When the robot is not operating, it can lower three legs to the ground to transition from dynamic stability to static stability. Here the robot stands stably, which allows the robot to charge its 4 12v sealed lead-acid batteries. These batteries allow the robot to run for approximately 1.5 hours.

Parameter	Symbol	Value
Z-axis CM from ball center	l	0.73 m
Ball radius	r	0.106 m
Ball mass	m_{ball}	2.44 kg
Roll moment of inertia about COM	I^b_{xx}	12.59 $\mathrm{k}gm^2$
Pitch moment of inertia about COM	I^b_{yy}	12.48 $\mathrm{k}gm^2$
Yaw moment of inertia about COM	I^b_{zz}	0.66 $\mathrm{k}gm^2$
Body mass	m_{body}	65.9 kg

Table 3.1: Ballbot Physical Properties

Balancing control is achieved using an industrial computer with an Intel Core processor running QNX Real-Time operating system. Real-time guarantees are necessary to assure that the robot maintains consistent control loop timing. This computer communicates with a VectorNav® VN-100 Inertial Measurement Unit (IMU) at 267 Hz to assess the robot's lean angle. The QNX machine sends commands to motor drivers controlling the IMB and yaw motors. A pair of 2-degree-of-freedom arms, seen in Fig. 3.1(b), are also controlled by this machine. These arms have series-elastic actuators, allowing for better bandwidth force control [Nagarajan et al., 2012b]. These simple arms can either have large brass masses on the ends to simulate the weight of more complex arms, or they can have rubber balls for pHRI studies. In this configuration, the balls can be attached in series with an Optoforce® three-axis force gauge.

The ballbot has another industrial computer onboard with a Core 2 Duo processor, running the Ubuntu Linux operating system. This machine communicates with the QNX machine, as well as many sensors and the ballbot's turret. One of the primary sensors is a Hokuyo UST-20LX scanning laser range finder. This range finder scans at 40 Hz, measuring ranges from .06 - 20 m.

A soft skin made from EVA foam was added to the ballbot in 2013. The foam is comprised of panels of thermoformed Kydex plastic with the foam bonded on top. The skin is segmented into panels for easy attachment via Velcro®. This skin helps facilitate HRI experiments, as people can touch or push the robot anywhere that the skin covers. The skin can be seen in Fig. 3.2(a).

Shown in Fig. 3.2(b), the ballbot's sensory turret has multiple sensors that facilitate human-robot interaction. These include an Asus Xtion point cloud sensor, a Hokuyo UST-05LN 5 m laser range finder, an Acoustic Magic directional microphone, and two speakers. The turret can

(a) The ballbot with turret, arms, and EVA foam skin

(b) The ballbot's pan/tilt turret with microphone, speakers, point clound sensor, and laser scanner

Figure 3.2: The ballbot with arms, skin, and sensory turret

pan $700°$, allowing for greater than a full rotation. It can tilt the point cloud sensor and laser range finder $90°$ to face straight down at the ground.

3.3 Dynamic Model

The work in this thesis relies on a dynamic model of the ballbot system. A 3-D model of ballbot was previously proposed and examined in Umanhankar Nagarajan's dissertation [Nagarajan, 2012]. Although this model is a better representation of the ballbot's dynamics than a 2-D planar model, it is very complex mathematically. As such, for the purpose of the derivations in Chapters 4 and 5, we used a two-dimensional representation of the ballbot. This means that the robot will be considered as two independent planar systems in the X and Y directions. As a result, any Coriolis coupling terms between these directions will not exist. Again, this is a less rich model than the 3-D case, but permits mathematical manipulation that enables the analysis and techniques in Chapter 4.

In practice, the differences between the full 3-D model and 2-D models are negligible for the purposes of the work presented. This is because the ballbot nominally operates with $\pm 5°$ lean angle and an average maximum $.5°$/s lean angular rate. Because the terms in the 3-D model that do not exist in the 2-D model are products of sine of the lean angle and angular rate, these correspond (when calculated in radians) to terms with $\approx 10^{-5}$ scale factors.

For the purpose of derivation and simplification, the 2-D ballbot model will be used, shown in Fig. 3.3. The model has two degrees of freedom: lean angle and ball angle with one control

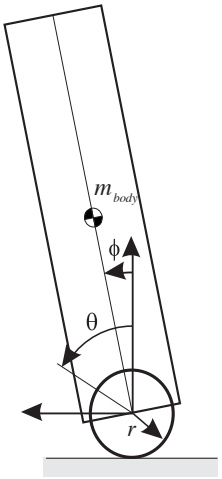


Figure 3.3: Planar ballbot model and notation diagram.

input, the torque between ball and body.

To find equations of motion, the Euler-Lagrange equation is used:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = \begin{bmatrix} \tau \\ 0 \end{bmatrix}.$$

Solving with state variables ϕ, the lean angle, and θ the ball angle, the equations of motion are:

$$M\ddot{q} + C\dot{q} + D + G = U, \tag{3.1}$$

with, the matrices as follows:

$$q = \begin{bmatrix} \theta \\ \phi \end{bmatrix}, M = \begin{bmatrix} I_{ball} + m_{body}r^2 + m_{ball}r^2 & -m_{body}lr\cos\phi \\ -m_{body}lr\cos\phi & m_{body}l^2 + I_{body} \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & m_{body}\dot{\phi}lr\sin\phi \\ 0 & 0 \end{bmatrix}, G = \begin{bmatrix} 0 \\ -m_{body}gl\sin\phi \end{bmatrix}, U = \begin{bmatrix} \tau \\ -\tau \end{bmatrix}.$$

The parameters of these equations are: height of the center of mass (from the center of the ball) l, radius of the ball r, mass of the sphere m_{ball}, moment of inertia of the sphere I_{ball}, mass of the body m_{body}, moment of inertia of the body I_{body}, and gravitational constant g.

The contact point of the ball with the floor is $x = r\theta$, so the ball angle should be considered as the position in space of the robot, for which the ballbot has direct actuation, while the lean angle is not actuated. D is the vector of frictional terms. For the purposes of trajectory generation and analysis, nonlinear friction is omitted from the model. U is the vector of control inputs. In this system the only control input is torque between body and ball. Eq. 3.1 yields two equations. Adding these two equations and replacing the second equation with the result yields:

$$M'(q)\ddot{q} + C'(q,\dot{q}) + G'(q) = \begin{bmatrix} \tau \\ 0 \end{bmatrix}. \tag{3.2}$$

The second equation from Eq. 3.2 is the internal system dynamics, equal to zero on the RHS.

3.4 Software and Control

3.4.1 Controller Design

The ballbot uses an inner and outer loop control scheme as discussed extensively in [Nagarajan et al., 2012a]. A PID controller is run on the desired lean angle, with an outer loop PD controller for trajectory following. This method has the advantage of placing limits on lean angle and

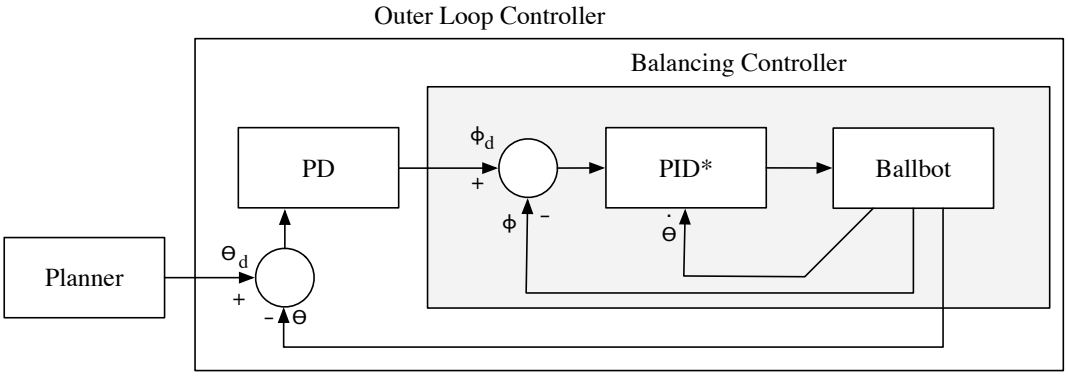


Figure 3.4: Ballbot Controller Diagram

ensuring that as long as the balancing controller is stable within the lean angle limits, the robot will not fall over. Although this may not yield the same performance as an inverse dynamics or LQR controller, having an inner loop balancer is a more robust approach as substantiated by experiments in [Nagarajan et al., 2012b]. As such, this method of control is utilized to test the feasibility of the generated trajectories. Maintaining balance is critical to the ballbots function, a task maintained by the inner loop balancer.

The ballbot's yaw motor is controlled with PID feedback control on the yaw estimate taken from the IMU. The turret pan and tilt are controlled via PID controllers with feedback from encoders. They are feed smooth quadratic trajectories in angle with a prescribed acceleration and a maximum velocity. These values have been set to an angular acceleration of $200°/s^2$ and a maximum angular velocity of $140°/s$ for panning. For tilt, the angular acceleration is set to $100°/s^2$ with a maximum angular velocity of $115°/s$.

3.4.2 Software Architecture

As discussed in Sec. 3.2, the software running on the ballbot is distributed between two machines: a real-time system and a high-level system. Some software also runs on a "base station" computer used to visualize data, command goals for the robot, monitor performance, and log data. A block diagram of these three systems can be seen in Fig. 3.5.

The QNX machine is responsible for balancing the robot and controlling the arms. The control loop is closed at 500 Hz, triggered by an RTC timer module. Tasks that are not real-time critical are run on the high-level machine. This makes coding more tractable, as software on this

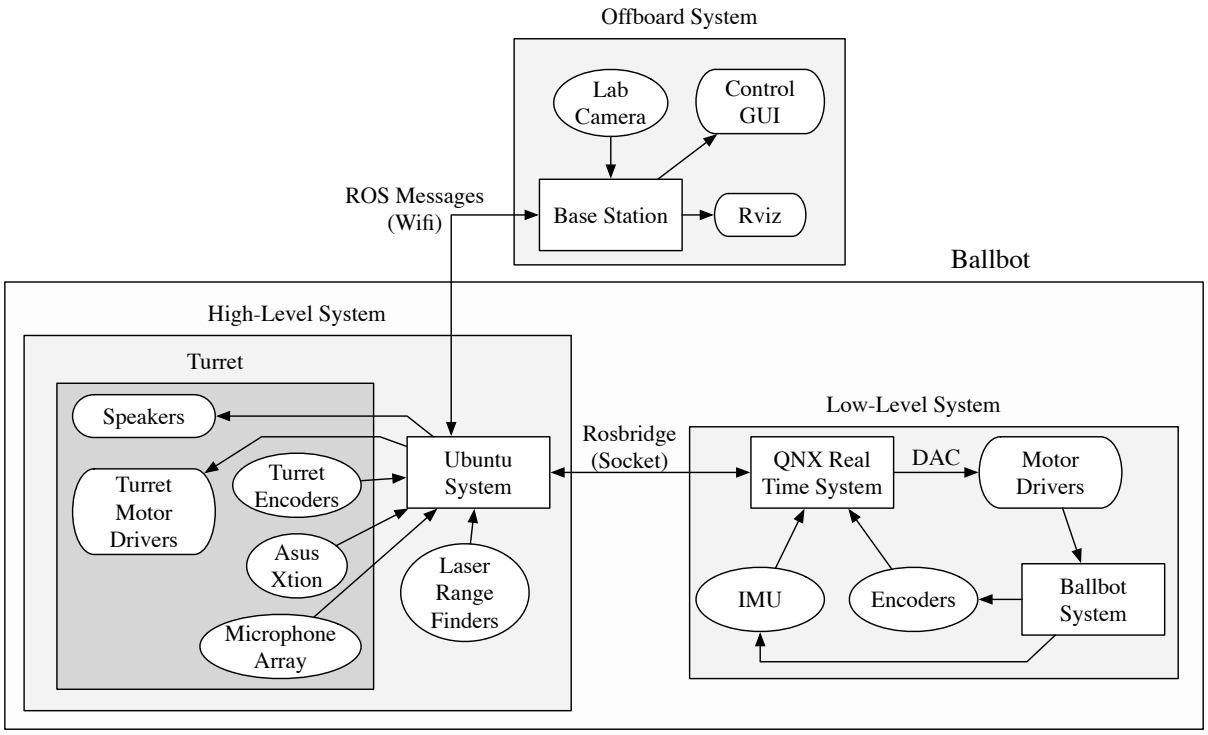


Figure 3.5: Ballbot Controller Hierachry

machine does not have to be real-time compliant. Software for this machine is written utilizing ROS [Quigley et al., 2009] Indigo. ROS is a meta-operating system for robots, facilitating communication between different pieces of code, and easy use of open-source components.

The two computers onboard the robot communicate using rosbridge [Crick et al., 2012], a serialization/deserialization package for ROS communication via socket connection. This is critical as pure ROS is incompatible with real-time systems, but rosbridge can work on both Ubuntu and QNX. The real-time system communicates odometry, lean angle, and diagnostic information to the high-level system. The Ubuntu machine sends desired lean angle, velocity, state-of-operation commands such as “Legs Up” and “Balance.”

The main components that run on the high-level machine are localization, navigation, and communication with the “base station.” Localization and mapping are accomplished via the open source package Hector SLAM [Kohlbrecher et al., 2011]. Hector SLAM utilizes the Hokuyo laser range finder to map and localize using scan-matching. The open source distribution has been modified for use on the ballbot to include the ability to save and load maps.

The high-level computer also communicates with a base station computer. This computer is used to command the robot into different states. This includes starting controllers, lifting the legs, starting to balance, etc. Dynamic reconfigure, a ROS component, is also used on this machine to tune gains and change parameters. The base station computer also runs a camera and visualization system for debugging and demonstration.

3.5 Experimental Setup

3.5.1 Laboratory Space

Although the ballbot is not limited to operation in any one particular place, the Microdynamic Systems Lab is where the vast majority of experiments take place. Located in Smith Hall 120 at Carnegie Mellon University in Pittsburgh, PA, the lab space houses ballbot and offers good space for experiments in navigation and HRI.

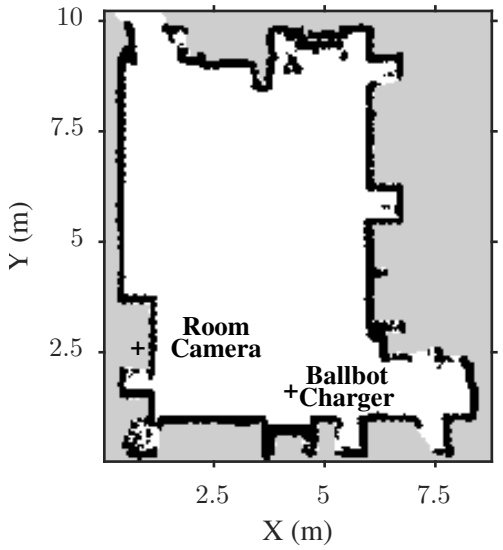


Figure 3.6: Top-down view of Smith Hall 120, where most ballbot experiments are run.

Seen in Fig. 3.6, the ballbot lab has approximately 6 x 7 m of open floor space that can be used for experimentation. Note that the map of the lab seen in Fig. 3.6 was generated via Hector SLAM. As will be shown in Chapter 5, this space allows for easy configuration of obstacles to test navigation strategies. The lab is also ideal for lab tours, offering enough space for groups of up to 20 people and a large TV for showing videos and live data visualizations with the robot.

3.5.2 Augmented Reality Visualization

The camera in Smith Hall 120, connected to the base station computer, is used to capture video of ballbot experiments as well as to show data overlays for debugging and demonstations. This is possible because of careful calibration and the use of ROS Rviz.

An example of the augmented reality (AR) system can be seen in Fig. 3.7. This figure shows the ballbot following a trajectory, shown in green. The red lines on boxes and wall are the laser scan returns. The laserscan is approximately 1 m from ground, mounted to the front of the ballbot. Overlaid on the ballbot's body are all the useful coordinate frames (using the ROS tf package). The red, green, and blue lines correspond to the x, y, and z directions. This particular

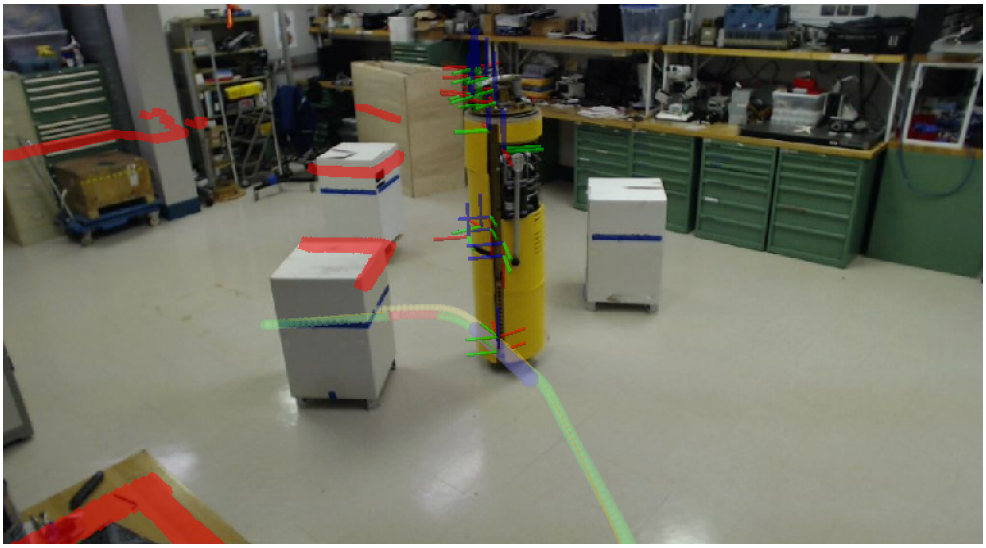


Figure 3.7: Example of the augmented reality system in Smith Hall 120. This view shows a trajectory and laser scan overlaid onto an image of the room from the camera. Also overlaid is a trajectory that the robot is currently following and all of the useful coordinate frames on the body. The view shown can be generated live on a televison in the lab for demonstration.

vantage point and system for overlay of data will be is used frequently in the document to present results of experiments.

Chapter 4

Flatness-based Trajectory Generation for the Ballbot

This chapter presents a method to find analytic and feasible trajectories for the ballbot. Navigation is a unique challenge with such a robot, as the planning and control are tightly coupled. Previous work discussed in Section 3.1 presented a method for ballbot navigation based on sequential composition of motion policies. Although this method has proven effective, it is restricted to the set of motions computed offline. Therefore, movements to arbitrary locations or recovering from poor initial conditions cannot be handled. This motivates the need for a method of trajectory generation that can quickly create feasible trajectories to and from arbitrary conditions. This chapter shows that this can be achieved through a relaxation of the dynamic equations and a transformation into a lower dimensional flat output space.

4.1 Differential Flatness

Differential Flatness[Fliess et al., 1995] is a property of some systems that can reduce the complexity of trajectory generation[Sira-Ramirez and Agrawal, 2004]. In defining a flat system, the goal is to find "flat output" variables, which are defined as functions of the state variables and their derivatives. There are as many flat outputs as control inputs to the system. For a flat output to be feasible, the state variables and control inputs must have a functional mapping from the flat outputs S and their derivatives:

$$S = Y(x, \dot{x}, \ddot{x}, ..., x^{(r)}) \tag{4.1}$$

$$x = A(S, \dot{S}, \ddot{S}, ..., S^{(\beta)}) \tag{4.2}$$

$$u = B(S, \dot{S}, \ddot{S}, ..., S^{(\beta+1)}). \tag{4.3}$$

In other words, if x is vector of state variables, the functions Y, A, and B must exist with finite r and β. If flat outputs exist for a given system, trajectory generation may become simpler, as trajectories of the flat output variables yield trajectories for the state variables and the control inputs. The flat output trajectory must simply have enough derivatives to satisfy the mapping back to the state variables and control inputs. In the case of a system like the planar ballbot,

there will be one flat output variable corresponding to the one control input. A trajectory in this variable will map to trajectories in lean angle and ball position. Therefore, if a flat output can be found, the task of generating a feasible trajectory will be reduced to simply finding a trajectory in the flat output with sufficient derivatives. In the general case, the flat output trajectories can have any basis. For the purpose of easily satisfying boundary conditions and having finite smooth derivatives, polynomials are used for this work.

4.2 Ballbot as a Flat System

As long as the flat outputs satisfy appropriate smoothness conditions, the outputs and their derivatives can be mapped back to the full state. To find flat outputs, the system must be linearized. Because the robot's lean angle is generally bounded $-5^\circ < \phi < 5^\circ$ in normal operation, it is reasonable to use the small angle approximation about equilibrium, *i.e.*, $\sin\phi \approx \phi$ and $\cos\phi \approx 1$, making the Eq. 3.2 become

$$\tau = \alpha\ddot{\theta} + (\alpha+\beta)\ddot{\phi} - \beta\phi\dot{\phi}^2 \tag{4.4}$$

and

$$0 = (\alpha+\beta)\ddot{\theta} + (\alpha+\gamma+2\beta)\ddot{\phi} - \beta\phi\dot{\phi}^2 + \frac{\beta g\phi}{r}, \tag{4.5}$$

with $\alpha = I_{\text{ball}} + (m_{\text{ball}} + m_{body})r^2$, $\beta = m_{\text{body}}rl$, $\gamma = I_{\text{body}} + m_{\text{body}}l^2$, each constants.

Next, to further linearize the system, the term $\beta\phi\dot{\phi}^2$ must be assumed negligible [Shomin and Hollis, 2013]. Intuition dictates that since ϕ should be nominally zero and $\dot{\phi}$ should be small, this term should be very small, as it is the product of three small quantities. With this assumption, Eq. 4.4 becomes:

$$\tau = \alpha\ddot{\theta} + (\alpha+\beta)\ddot{\phi} \tag{4.6}$$

and Eq. 4.5, the internal dynamics, becomes:

$$\underbrace{r(\frac{\alpha}{\beta}+1)}_{\lambda_1}\ddot{\theta} + \underbrace{\frac{r}{\beta}(\alpha+\gamma+2\beta)}_{\lambda_2}\ddot{\phi} = g\phi, \tag{4.7}$$

$$\lambda_1\ddot{\theta} + \lambda_2\ddot{\phi} = g\phi. \tag{4.8}$$

λ_1 and λ_2 are constants. With this equation, the flat output S is defined as:

$$S = \lambda_1\theta + \lambda_2\phi, \tag{4.9}$$

with derivatives:

$$\dot{S} = \lambda_1\dot{\theta} + \lambda_2\dot{\phi}, \tag{4.10}$$

$$\ddot{S} = \lambda_1\ddot{\theta} + \lambda_2\ddot{\phi} = g\phi \tag{4.11}$$

from Eq. 4.8, and

$$\dddot{S} = g\dot{\phi} \tag{4.12}$$

$$S^{(4)} = g\ddot{\phi}. \tag{4.13}$$

This mapping is chosen after recognizing the relationship between ϕ, $\ddot{\theta}$, and $\ddot{\phi}$ in Eq. 4.8. This gives the mapping of the two state variables θ and ϕ to the flat output variable S in Eq. 4.9. Four derivatives of S are shown because these are required to get the inverse mapping to the state variables and their derivatives present in Eqs. 4.4 and 4.5. For the flat output to be valid, there must be a mapping to the control input τ. Because $\ddot{\phi}$ appears in the equation for τ, a mapping to $\ddot{\phi}$ is required. Leaving g on the right hand side, outside of the λ constants emphasizes that the system is shape accelerated, meaning it's acceleration comes from nonzero lean angle. It should be noted that with these four derivatives of S, there is a mapping from both the state variables and a finite number of derivatives to S and a mapping from S and a finite number of derivatives to the state variables and the control input. Therefore, S is a flat output of the linearized system in Eqs. 4.6 and 4.7. Using this relation, trajectory generation can now be simplified.

4.2.1 Trajectory Generation

The ballbot is accelerated by its lean angle ϕ, but the more important state variable in high level planning is the ball angle θ, as it can be thought of as the global position of the robot. From the previous section, it has been shown that from a flat output trajectory, the feasible θ trajectory can be found for the linearized system in Eqs. 4.6 and 4.7. In the control law derived in the next section, it will be shown that τ depends on the fourth derivative of S; as such S needs to be at least C^4 for a point-to-point motion, with desired initial and final configurations, making ten conditions. With these constraints and the choice of a polynomial basis, a nonic (ninth order) polynomial is needed. After constructing this polynomial S_d from the constraints, the state variable trajectories are simply found by solving Eqs. 4.9 and 4.11 for θ and ϕ:

$$\theta^* = \frac{1}{\lambda_1}\left(S_d - \frac{\lambda_2}{g}\ddot{S}_d\right) \tag{4.14}$$

$$\phi^* = \frac{\ddot{S}_d}{g}. \tag{4.15}$$

Given a smooth function in S (in this case a nonic polynomial), a feasible trajectory in θ^* and ϕ^* can be easily calculated which satisfies the internal system dynamics. Because there is a mapping in the opposite direction (Eq. 4.9), the S trajectory can be constructed with initial and final conditions given by state variables. Fig. 4.1 shows an example of a rest-to-rest motion and feasible θ trajectory that comes out. S has been rescaled by λ_1 for the purpose of comparison. This is a rather agressive trajectory, moving the robot two meters in three seconds. As such, a very large lean angle is required. As can be seen from the θ output trajectory in Fig. 4.1, the ball must first roll back before moving forward to achieve that lean angle. It then overshoots and comes back to achieve the lean back required for deceleration. The fact that this behavior comes out naturally from the flat output trajectory is a significant advantage of this method. The polynomial in S was constructed from initial conditions of $\theta_0 = 0, \dot{\theta}_0 = 0, \phi_0 = 0, \dot{\phi}_0 = 0, \ddot{\phi}_0 = 0$ and final conditions $\theta_f = rx_f = 2r \approx 24\,\mathrm{rad}, \dot{\theta}_f = 0, \phi_f = 0, \dot{\phi}_f = 0, \ddot{\phi}_f = 0$.

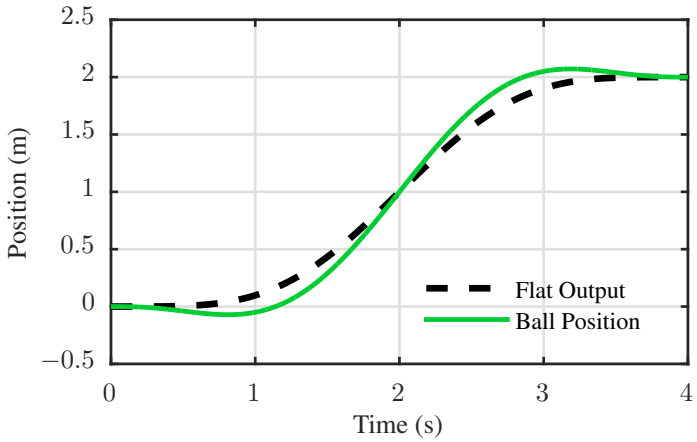


Figure 4.1: Desired flat output trajectory and corresponding feasible θ trajectory. Note that the flat output is scaled by $1/\lambda_1$ for comparison to θ

4.2.2 Trajectory Optimization

As stated previously, the flat output trajectory is a nonic polynomial, and in the previous example shown in Fig. 4.1, ten constraints were used to generate it. However, for many cases, it may be beneficial to leave some of those parameters free. In a quick stopping trajectory, for example, the final position may not be important, but coming to zero velocity quickly is important. As seen in quadrotor trajectory generation[Mellinger and Kumar, 2011], smooth, well-formed, (in the sense of the control input) trajectories can be found by minimizing the snap (fourth derivative of position) over the trajectory. The intuition for this cost function comes from Flash and Hogan [Flash and Hogan, 1985] who found that human arm trajectories minimize jerk (third derivative of position). The quadrotor model control laws contained one higher derivative than the human arm model, so instead of jerk, snap was minimized. The planar ballbot model contains one higher derivative still: $S^{(4)}$, the snap of the flat output, which appears in the control law. As such, the cost function used in this work finds feasible trajectories with free parameters by minimizing $S^{(5)}$, the crackle of the flat output. Note also that $S^{(5)}$ is proportional to $\dddot{\phi}$, so this is equivalent to minimizing the jerk of the lean angle, which has been found empirically to produce good trajectories in previous work [Nagarajan et al., 2012a].

Minimizing $S^{(5)}$ over the whole trajectory can be formulated as a quadratic program (QP), since S is simply a ninth order polynomial:

$$\begin{aligned} \min \quad & c^T H c \\ \text{s.t.} \quad & Ac \leq b. \end{aligned} \tag{4.16}$$

where c is a vector of the polynomial coefficients and non-free constraints become equality constraints $Ac = b$ in the QP. H is a positive definite matrix and as such, the QP can be solved in polynomial time. This allows for real time, optimal trajectory generation online. The ballbot updates its controller at 500 Hz, and using this formulation, the QP can be solved in less than 1 ms, allowing for trajectory generation in two orthoganol directions in one controller update step.

One huge advantage of the flat output trajectories is that they can be scaled spatially and temporally since they are simply polynomials, compared with numerical trajectories which have no such property. This means that any safety requirements such as a maximum lean angle can be satisfied by lengthening the time of a trajectory, since all the derivatives of S go to zero as t goes to infinity. When temporally rescaled, derivatives of S simply have a scale factor of $(t_{old}/t_{new})^k$ where t_{new} is new time length of the trajectory, t_{old} is the original, and k is the derivative to be scaled. Note that this scales the initial and final conditions (other than the zero derivative) if they are nonzero. This means that the time length of the trajectory must be known *a priori* if the initial or final derivatives are nonzero. Scaling time is also useful for numerical stability in solving the QP. Since S is a nonic polynomial in t, t^9 appears in the minimization. As such, using unit time of $0 < t < 1$ keeps the numerical solution from going unstable.

4.2.3 Rest-to-Rest Motions

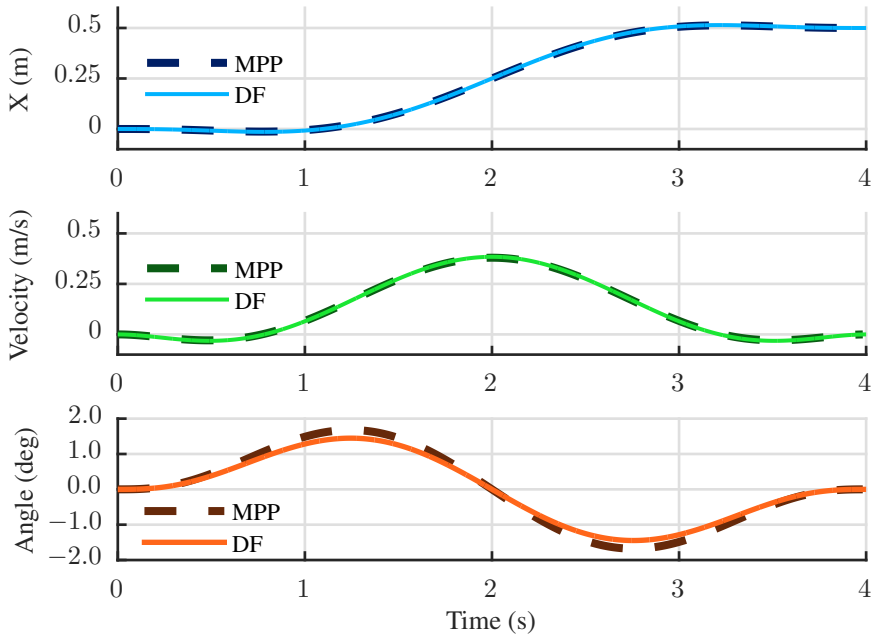


Figure 4.2: Comparison of Differentially Flat (df) Trajectory with corresponding trajectory found via offline nonlinear minimization for Motion Policy Primitives (MPP)

Possibly the most useful for normal navigation, rest-to-rest motions are trajectories which begin and end with all flat output derivatives equal to zero. Note that to achieve forward motion, the ball first must roll backwards to achieve a positive lean angle. This simulated motion is the same ball trajectory shown in Fig. 4.1. Rest-to-rest motions were previously explored in [Nagarajan et al., 2009b] and [Nagarajan et al., 2012a]. A comparison of the differentially flat approach to the approach in [Nagarajan et al., 2012a] can be seen in Fig. 4.2. The previous approach utilized minimization of the deviation from the nonlinear internal system dynamics. Ball position

and velocity are within 1% for the whole trajectory while the lean angle trajectory differs by a maximum of 12%. This difference could be attributed to the differences between the linearized and nonlinear system. The key difference between these methods is that the differentially flat trajectory can be computed in less that 1 ms, allowing for online generation, unlike nonlinear minimization which must be done offline.

4.2.4 Arbitrary Motion and Free Constraints

Differentially flat trajectory generation scales to more complex situations than rest-to-rest motions. Examples include recovering from an outer loop controller going unstable or significant disturbances. Generating a feasible trajectory in a single controller time step before applying torque to recover allows for feasible feedforward terms. This means less vulnerability to higher order modes becoming excited. One current drawback of the method is that the length of time the trajectory will take must be known a prior to satisfy the initial conditions. In practice, it has been found that trajectories of 4 s or less can recover from any reasonable condition without exiting the safe domain of the balancer. In situations when recovery is important but the final position is not, the constraints on the final ball angle are left free. This is a significant advantage of the QP formulation, as it can still construct the polynomial within one time step, but also find the best smooth path that minimizes lean angle jerk without regard to final location.

4.3 Experimental Validation

This section presents an experimental validation of the flatness-based trajectory generation method. Firstly, rest-to-rest motions for distances of 0.5 m to 2.0 m were run with the robot. Time was varied to affect the maximum lean angle achieved in the trajectory. Maximum angles from 1.2° to 5° were achieved. One such trajectory can be seen Fig. 4.3.

All of the trajectories attempted were run open loop, using only the feedforward lean angle for the balancer. This was done simply to show this method is a feasible means to create ballbot trajectories. Running the motions open-loop also shows that model for the system is adequate, as the tracking is still acceptable without feedback. This shows that the trajectories are feasible for the ballbot. For the trajectory shown in Fig. 4.3, the maximum velocity tracking error is 15% with the trajectory slightly undershooting, but ending 10% short of the goal. Note that results presented in Chapter 5 utilize full state feedback and the outer-loop controller shown in Section 3.4.2

Feasibility of recovery trajectories was assessed by testing recovery from both an initial lean angle and an initial velocity. A successful recovery from a 2° lean angle can be seen in Fig 4.4. The lean angle was slowly ramped up to 2° while the robot was held in place. The robot was let go, and it began to accelerate. After being let go, the trajectory planner generated a stopping trajectory and executed it. This was attempted 6 times, all successful recoveries. Recovery from initial velocity was also tested by pushing the robot and generating a stopping trajectory. This was also successful in bringing the robot safely to a stop from initial velocities up to 1 m/s.

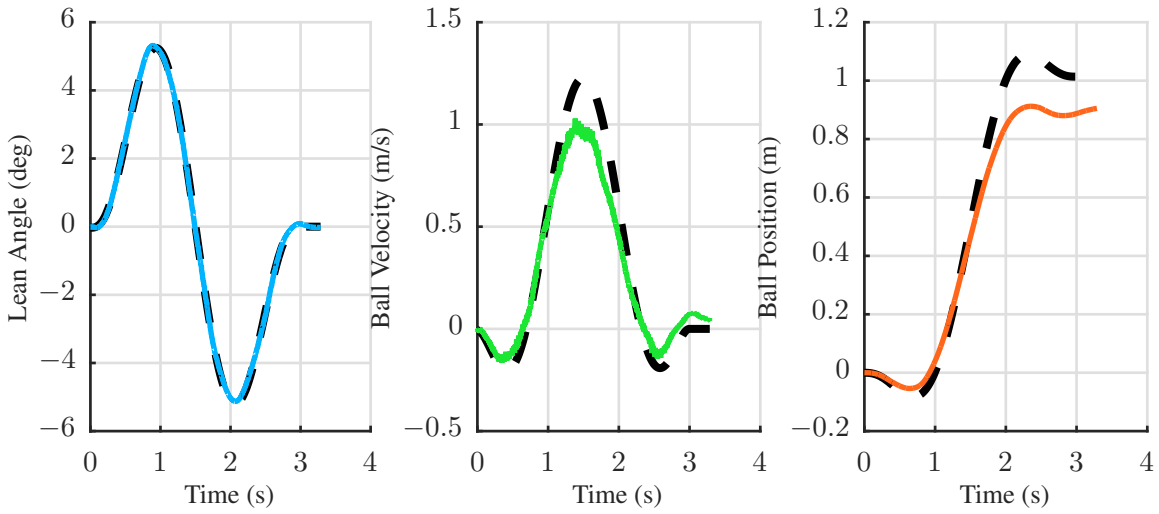


Figure 4.3: Lean angle, velocity, and position data from a rest-to-rest motion. Desired trajectories are shown in dashed black with the colored lines showing actual data. This experiment was run open loop with only feedforward lean angles given to the balancing controller.

4.4 Summary

The chapter presented a method of fast, analytic trajectory generation for dynamically stable balancing robots. Relaxing the equations of motion to form a differentially flat system allows for analytic generation of feasible trajectories. Furthermore, the method has been extended to use free constraints for recovering from poor initial conditions. This method was experimentally verified with the ballbot, a single-wheeled balancing system. The derivation of this technique also confirmed some empirically-found notions from previous work [Nagarajan et al., 2012a]. These include using a nonic polynomial for trajectories and minimizing the jerk of lean angle.

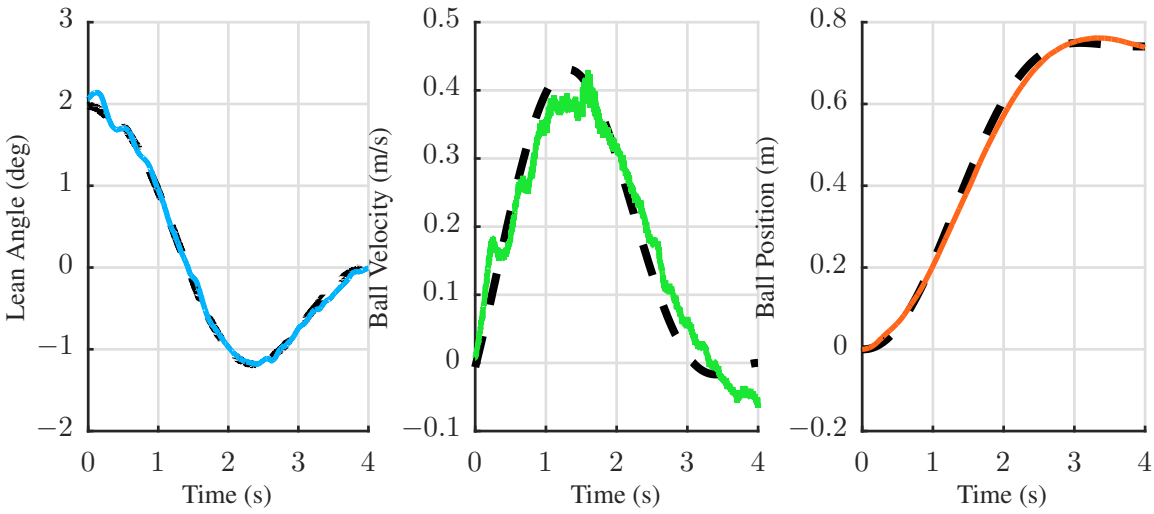


Figure 4.4: Lean angle, velocity, and position data from a recovery motion. Desired trajectories are shown in dashed black with the colored lines showing actual data. This experiment was run open loop with only feedforward lean angles given to the balancing controller.

Chapter 5

Ballbot Navigation via Flatness

This chapter extends the work of Chapter 4 to multi-waypoint trajectory optimization for the ballbot. The capability enables the ballbot to navigate among obstacles, both static and dynamic. Planning for dynamic robots generally requires a higher dimensional plan than for kinematic robots, as the dynamics of the robot must be respected in the trajectory planning. Kinodynamic planning [Donald et al., 1993] tries to solve the problem of concurrent dynamic and kinematic plans. This still requires a high dimensional search space, and as such takes considerable computational effort when attempted with techniques such as graph search. Most efforts attempt to sidestep this bottleneck by using techniques such as RRT* [Webb and van den Berg, 2013]. Such efforts rely on controllable linear dynamics, however, and still need to search a high dimensional space, often making them infeasible for real-time operation. The work presented in this chapter instead uses the formulation of the ballbot as a differentially flat system, making generation of feasible trajectories much more straightforwad.

5.1 Multi-Polynomial Optimization

Formulation of the ballbot as a flat system enables very fast generation of feasible polynomial trajectories. This section shows how a system was developed to use these polynomials to generate multi-segment trajectories that can navigate cluttered environments. These trajectories respect the full dynamics of the system and can still be computed very quickly.

Continuing from the previous chapter, consider $p(t)$ as the nonic polynomial in S with coefficients c. In the joint optimization of all polynomial trajectories through a set of waypoints, c represents a column vector of all the coefficients of the sequential polynomials in the trajectory. These polynomials must be consistent at the intermediate waypoints, meaning that the boundary conditions of the consecutive polynomials must be equal at their shared waypoints. As per [Shomin and Hollis, 2014], the sum squared crackle of the polynomials can be written as

$$\int_{t_0}^{t_f} (c^T p_5(t))^2 \mathrm{d}t = \bar{c}^T H \bar{c}, \tag{5.1}$$

where $p_5(t)$ is the fifth derivative (crackle) of the flat output.

Since we are interested in the 5th derivative (crackle) of $p(t)$:

$$p_5(t) = \begin{bmatrix} \frac{9!}{4!}t^4 & \frac{8!}{3!}t^3 & \frac{7!}{2!}t^2 & 6!t & 5! & 0_{1\times 5} \end{bmatrix}^T \tag{5.2}$$

where $0_{1\times 5}$ is a 1 by 5 vector of zeros. With this formulation, the problem can be posed as the minimization of the sum squared crackle from t_0 to t_f, the start and end times of the polynomial:

$$\min \int_{t_0}^{t_f} (c^T p_5(t))^2 \mathrm{d}t. \tag{5.3}$$

Now let a be just the nonzero coefficient from (5.2), and let $\bar{c}$ be the associated polynomial coefficients: c_9 - c_5. Then the objection function becomes:

$$\bar{c}^T \mathrm{diag}(a) \begin{bmatrix} t^8 & t^7 & t^6 & t^5 & t^4 \\ t^7 & t^6 & t^5 & t^4 & t^3 \\ t^6 & t^5 & t^4 & t^3 & t^2 \\ t^5 & t^4 & t^3 & t^2 & t \\ t^4 & t^3 & t^2 & t & 1 \end{bmatrix} \mathrm{diag}(a)\bar{c}. \tag{5.4}$$

After integration and evaluation from t_0 to t_f, the center matrix of t's becomes:

$$\begin{bmatrix} t_e(9) & t_e(8) & t_e(7) & t_e(6) & t_e(5) \\ t_e(8) & t_e(7) & t_e(6) & t_e(5) & t_e(4) \\ t_e(7) & t_e(6) & t_e(5) & t_e(4) & t_e(3) \\ t_e(6) & t_e(5) & t_e(4) & t_e(3) & t_e(2) \\ t_e(5) & t_e(4) & t_e(3) & t_e(2) & t_e(1) \end{bmatrix} = T, \tag{5.5}$$

where $t_e(i) = \frac{1}{i}(t_f^i - t_0^i)$. Now, let $H = \mathrm{diag}(a)T\mathrm{diag}(a)$ and it can be seen that the integral of sum squared crackle, (6.7), can be written as

$$\int_{t_0}^{t_f} (c^T p_5(t))^2 \mathrm{d}t = \bar{c}^T H \bar{c}. \tag{5.6}$$

(5.6) is now in the form of a cost function for a quadratic program (QP) that minimizes the squared crackle over a trajectory. To use the whole vector c, the cost matrix needs to have zero padding:

$$Q = \begin{bmatrix} H & 0_{5\times 5} \\ 0_{5\times 5} & 0_{5\times 5} \end{bmatrix}. \tag{5.7}$$

Now the optimization becomes:

$$\min \quad c^T Q c \quad \text{s.t.} \quad Ac = b. \tag{5.8}$$

Here the equality constraint is the position of the desired waypoints and any fixed derivatives. This formulation extends to multi-waypoint trajectories naturally by concatenating the polynomial coefficients in c of multiple consecutive segments. Q is then repeated along the diagonal to

create a block diagonal matrix [Mellinger and Kumar, 2011]. In this case, the equality constraints must also enforce consistency, meaning that the derivatives at the end of one polynomial segment must be equal to derivatives at the start of the next segment. This formulation will produce multi-waypoint minimum crackle trajectories, but it is numerically unstable for large trajectories. This is because the decision variables, the polynomial coefficients, can be of vastly different magnitudes. This failure can be seen in Fig. 5.2(b). The figure shows that although the first quarter of the trajectory is optimized properly, eventually the solver can no longer satisfy the consistency constraints and the path is discontinuous. Fortunately, the problem can be formulated as a much more stable, unconstrained QP.

5.1.1 Unconstrained Optimization

The key insight to reformulating the optimization is that the problem can be formed as an equivalent QP, but with the waypoint positions and derivatives as the decision variables instead of the polynomial coefficients. Constructing the problem this way both removes the equality constraints and uses decision variables bounded to a much smaller range. This change of variable has actually already been computed in (5.8). The A matrix in this equality constraint converts the coefficients c to the positions and derivatives, b. However, these are expressed in terms of the flat outputs, and it is more desirable to express these parameters in terms of the state d. Using d as the decision variable, the cost function can be reformulated as the total cost J:

$$J = d^T C A^{-T} Q A^{-1} C^T d, \tag{5.9}$$

where C is a selector matrix of ones and zeros which can rearrange the order the decision variables in the vector d. This derivation of the cost function is identical to its original derivation for quadrotors Richter et al. [2013]. The key difference is that quadrotors are trivially differentially flat, in that their state variables are the flat outputs. This is why the transformation matrix C can be only zeros and ones. For ballbot, or any other nontrivially flat system, another transformation matrix is required to go from polynomial coefficients to state variables. From (4.9), the necessary transformation M can be found as:

$$Ac_s = b, Ac_s = Md \tag{5.10}$$

$$c_s = A^{-1}Md \Rightarrow d^T M^T A^{-T} = c_s^T. \tag{5.11}$$

where M is a block diagonal matrix with matrices m along the diagonal:

$$m = \begin{bmatrix} \lambda_1 & 0 & \lambda_2 & 0 & 0 \\ 0 & \lambda_1 & 0 & \lambda_2 & 0 \\ 0 & 0 & g & 0 & 0 \\ 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & g \end{bmatrix}. \tag{5.12}$$

M is now a matrix that transforms flat outputs and derivatives: $b = [S\dot{S}\ddot{S}S^{(3)}S^{(3)}]^T$ to state variables: $d = [\theta\dot{\theta}\phi\dot{\phi}\ddot{\phi}]^T$. It is worth noting that A in (5.10) and its inverse can be solved in closed form as a function of the segment time. With this new transformation M, (5.9) becomes

$$J = d^T R d, \tag{5.13}$$

where R is

$$R = CM^T A^{-T} Q A^{-1} M C^T. \tag{5.14}$$

This matrix can be partitioned into fixed and free derivatives along with the vector of ballbot states d Richter et al. [2013]:

$$R = \begin{bmatrix} R_{FF} & R_{FP} \\ R_{PF} & R_{PP} \end{bmatrix}, d = \begin{bmatrix} d_F \\ d_P \end{bmatrix}. \tag{5.15}$$

d_f is the vector of fixed derivatives. Formulated as such, the optimal free waypoint derivatives, d_p^*, can be solved for as:

$$d_p^* = -R_{PP}^{-1} R_{FP}^T d_F. \tag{5.16}$$

This unconstrained formulation is both more numerically stable and of lower dimension than the constrained formulation. A comparison to the constrained method can be seen in Fig. 5.2(c). This trajectory optimizes over 44 waypoints. The first and last waypoints are fully constrained as they are set to be at rest with zero velocity and lean angle. The 42 interior waypoints are constrained only in position, while the other derivatives are free to be optimized. This example was solved in MATLAB using gaussian elimination on an Intel Core I7 machine in 1.1 ms. Conversely, the constrained system in Fig. 5.2(a) took 76 ms and still diverged. The unconstrained method is also very stable, with trajectories up to 250 waypoints being tested. An example of such a trajectory is shown in Fig. 5.1. The figure shows this trajectory generated in a random gridworld costmap, as will be discussed in the next section.

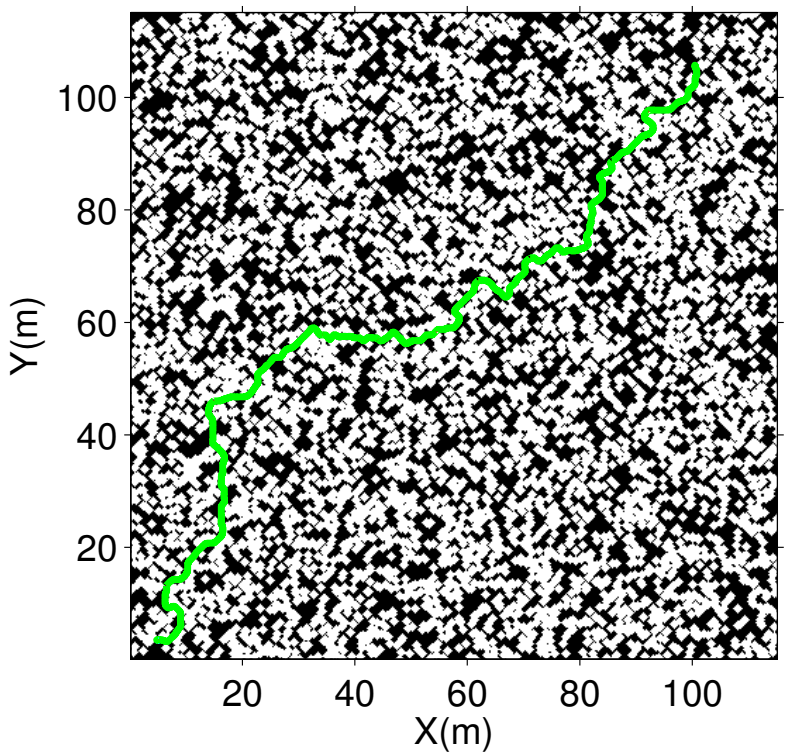
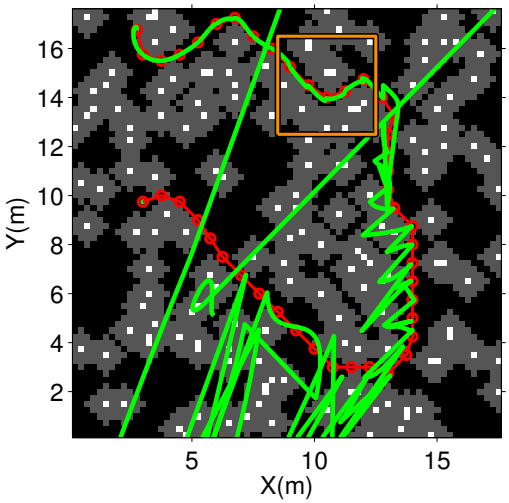
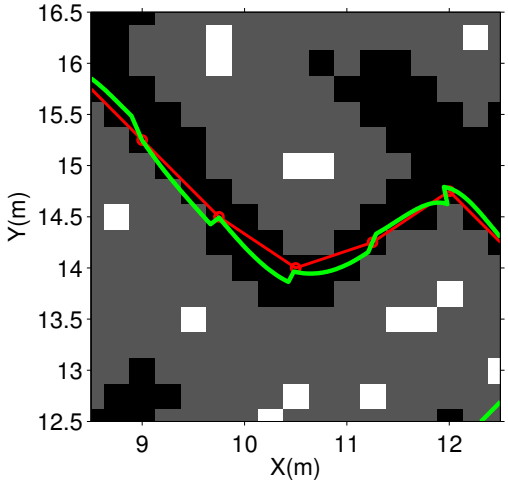


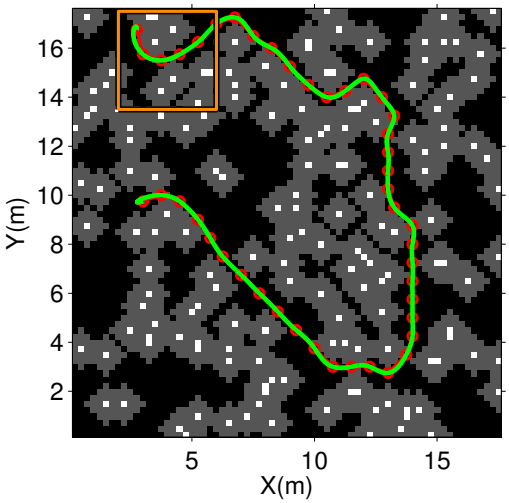
Figure 5.1: A very long path, planned and optimized. The trajectory is generated through over 250 waypoints and over 100 m. This shows the stability and power of the unconstrained optimization formulation. Black represents free space while white areas are randomly generated obstacles. The green path is the optimized trajectory connected a randomly selected start and goal.



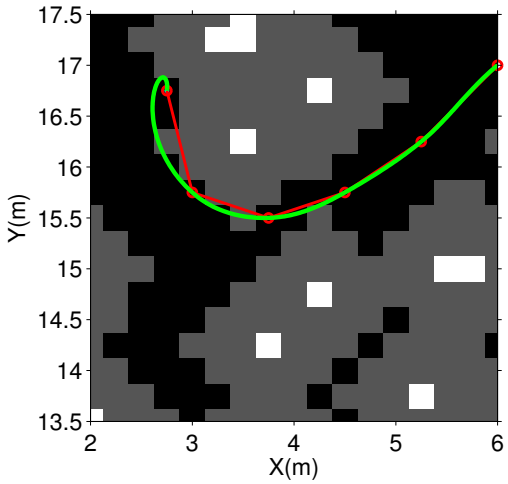
(a) Interior Point QP Solver solution for the constrained problem



(b) Magnified view of orange box from (a)



(c) Unconstrained QP solved as a linear system

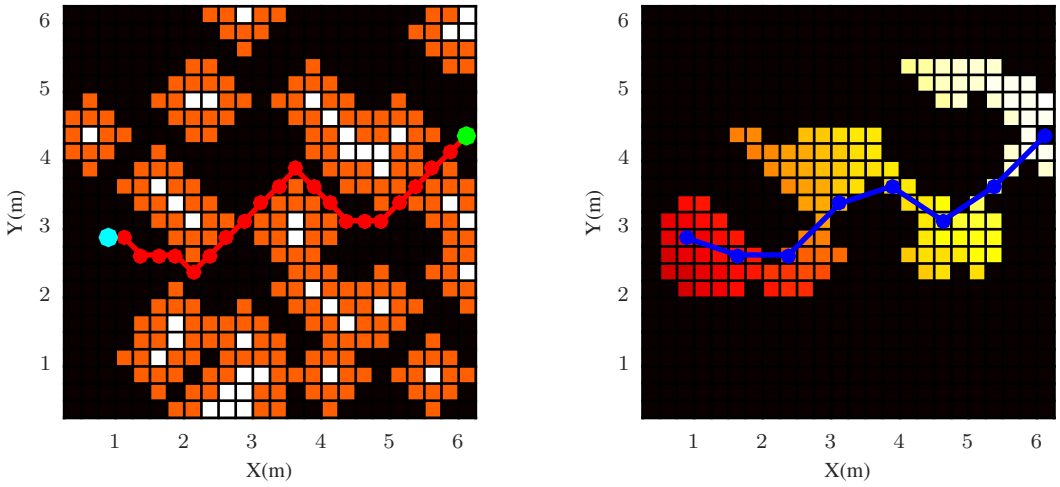


(d) Magnified view of orange box from (c)

Figure 5.2: Optimization of the same kinematic path (shown in red) using different techniques. Free space is shown in black, obstacles in white, and obstacle inflation in gray. Because this trajectory has 44 waypoints, the unconstrained optimization shown in (c) and (d) is the only successful method yielding a feasible dynamic trajectory.

5.2 Costmap Integration

As discussed in the previous section, a feasible trajectory for the ballbot can be generated from a series of waypoints; however, this does not take obstacles into account. To this end, a graph search planner is used to generate a kinematic, collision-free trajectory. That trajectory is then subsampled into waypoints to optimize a minimum crackle trajectory which is dynamically feasible by the strategy presented in the previous section. An example of this can be seen in Fig. 5.3.



(a) A* generated optimal kinematic path through a gridworld with random obstacles. Obstacles are white with orange inflation. The green dot is the start and the cyan dot, the goal. The red path is the output of A*.

(b) The colored cells represent the states expanded by A* in the graph search. Their color represent the value calculated by A*. The blue points are the waypoints, subsampled from the red path in (a).

Figure 5.3: An optimal kinematic path is generated through a gridworld using A*. The cell size is .25 m for the purpose of demonstration of the method.

Fig. 5.3 shows a gridworld with .25 m grid size and randomly generated obstacles shown in white. A*[Hart et al., 1968] is used to find an optimal kinematic path shown in red. This kinematic path is not dynamically feasible for ballbot, but does provide a good starting point, as it successfully avoids all obstacles in a path to goal. In this example, obstacles are shown in white and their inflation shown in orange. Fig. 5.3(b) shows the expanded states of A* and a subsampled path from the red initial path. The set of states shown in blue are used as waypoints to generate a dynamically feasible trajectory.

This example is continued in Fig. 5.4, which shows the dynamically feasible path generated for the ballbot. The green path shows the ball trajectory, while the red path is the generated center of mass trajectory. Note that the deviation between these paths can be used to visually infer the plan lean angle direction. This method is very powerful, as the graph search is done in a two dimensional space, instead of the 8 dimensional full state space. However, this method requires a conservative kinematic plan. Namely, Fig. 5.4 shows that the ballbot trajectory passes through the obstacle inflation in multiple locations. To ensure that the deviation of the dynamically

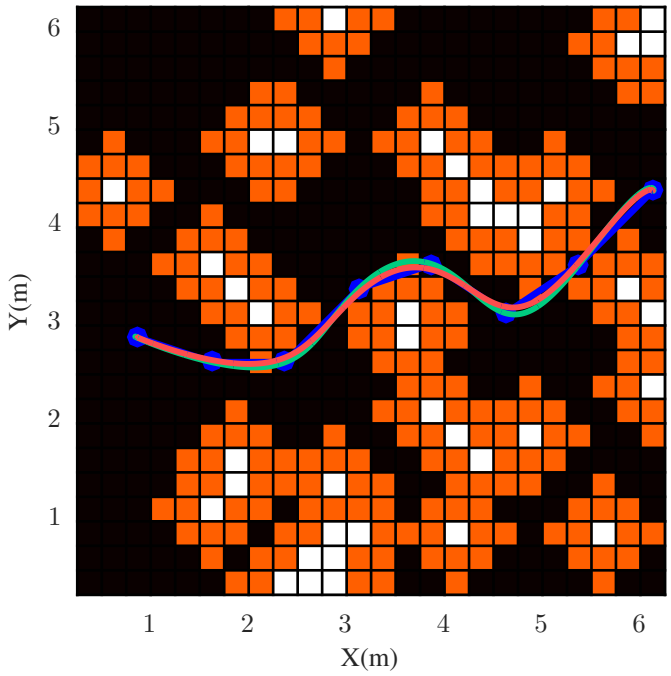


Figure 5.4: Continued from Fig. 5.3, the subsampled waypoints are used to generate an optimal feasible trajectory. The green trajectory shows the planned motion of the ballbot's ball, passing through each waypoint. The red trajectory shows the planned motion of the robot's center of mass.

feasible path from the kinematic path does not hit an obstacle, the obstacles must be inflated by approximately the distance between subsampled waypoints.

This example was generated in MATLAB and is useful for illustrating how the algorithm works, but the grid cells are far too coarse for actual planning. The system implemented on ballbot uses the ROS navigation stack to generate kinematic trajectories on costmaps with a grid resoluation of .05 m instead of .25 m. Fig. 5.5 shows an example of a trajectory generated in Smith Hall. In this example, obstacles are black grid cells, with an inflation shown as blue cells.

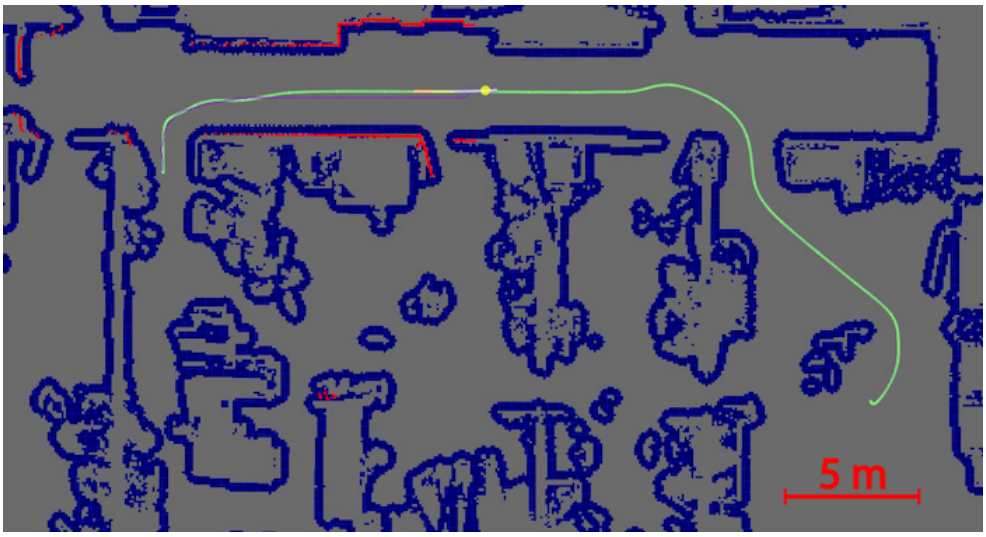
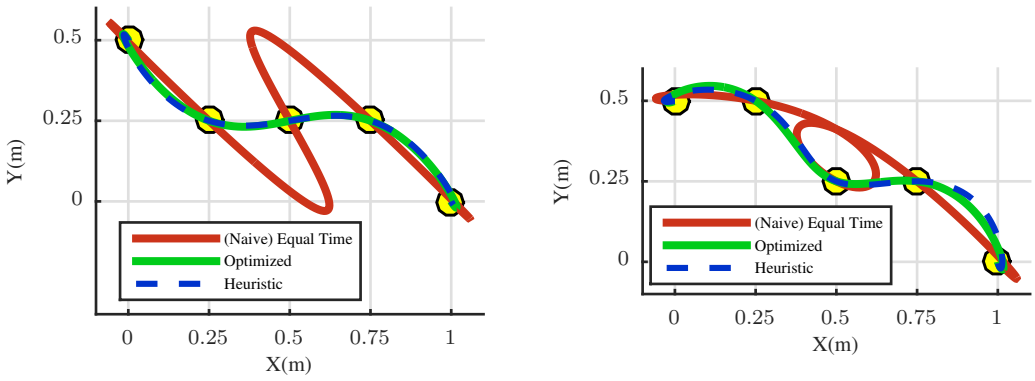


Figure 5.5: Top view of ballbot executing a 25 m trajectory through a building. The desired path is shown in green; laser scanner returns are shown as red dots; Inflated obstacles are shown in blue, and the current ball position is shown as a yellow dot.

5.3 Polynomial Segment Time-Allocation Optimization

Minimizing the energy of a series of polynomial segments, as shown in the Section 5.1.1, relies on knowing how much time each segment should take. From (5.5), it is clear that t_f and t_0 must be known for each polynomial segment before constructing the T matrices. The choice of these segment switching times is very important, as poorly allocated times will impose unwanted constraints on the robot motion, as shown in Fig. 5.6(a).



(a) Example #1 of trajectory through five waypoints using different segment time allocation.

(b) Example #2 of trajectory through five waypoints using different segment time allocation.

Figure 5.6: Comparison of the optimized trajectories using different methods of time allocation. Red - Equal times (naive), Green - Optimized times, Blue Dashed - Heuristic

In this figure, a trajectory is generated which is constrained to take a total of 5 s and pass through the 5 yellow waypoints, starting and ending at rest. The red path is the trajectory generated by naively choosing equal times, 1.25 s, for each segment. The green path shows a trajectory with times that were optimized by iteratively solving the QP and using gradient descent to yield a locally minimal energy [Richter et al., 2013]. The blue dashed path uses no time optimization, but instead allocates time based on a heuristic. The unoptimized trajectory in Fig. 5.6(a) (red) requires a maximal lean angle of $4.07°$, whereas the optimized trajectory (green) requires only a $1.63°$ maximum lean angle. For the ballbot, this corresponds to an instantaneous acceleration of .82 and .33 m/s^2, respectively. Note also that these trajectories take the same total amount of time. The trajectory which has times allocated by the heuristic has a maximum lean angle of $1.67°$, slightly more than the optimized trajectory. Although the optimized trajectory has lower required acceleration, using the heuristic has two major advantages: it requires no extra optimization step, and it is not susceptible to local minima.

Fig. 5.6(b) shows another example with the same three methods for time allocation, but slightly different waypoint positions. The difference is even more extreme here, as the naive allocation makes a loop that would be highly undesirable in execution.

This heuristic (blue dashed lines) calculates times to satisfy the dynamics of a constant acceleration system with a capped maximum velocity. The following algorithm details the time

allocation for the polynomial segment times, t_i. The algorithm uses v_m, a specified maximum velocity; a, the specified acceleration; d_s the polynomial segment Euclidean distance; v_0 the segment's inital velocity; and v_f, the segment's final velocity. This heuristic operates as follows:

Algorithm 1 Time-Allocation Heuristic

```
Assign waypoint velocites from kinematic plan
for all segments do
    t_1 ⇐ |v_m − v_0|/a
    d_1 ⇐ ((v_0 + v_m)/2) ∗ t_1
    t_2 ⇐ |v_m − v_f|/a
    d_2 ⇐ ((v_f + v_m)/2) ∗ t_1
    if d_1 + d_2 < d_s then
        t_m ⇐ (d_s − (d_1 + d_2))/v_m
        t_i ⇐ t_1 + t_m + t_2
    else
        t_i ⇐ t_1 + t_2
    end if
end for
```

In short, this heuristic assigns time to polynomial segments as if the ballbot were a second-order system with instantly attainable bounded acceleration and maximum velocity. This corresponds to piecewise linear velocity profiles between waypoints. Clearly, ballbot is not such a system, but the optimization of the trajectory takes care of that. The time allocation, however, is close enough to this system that no iterative optimization of segment time is necessary.

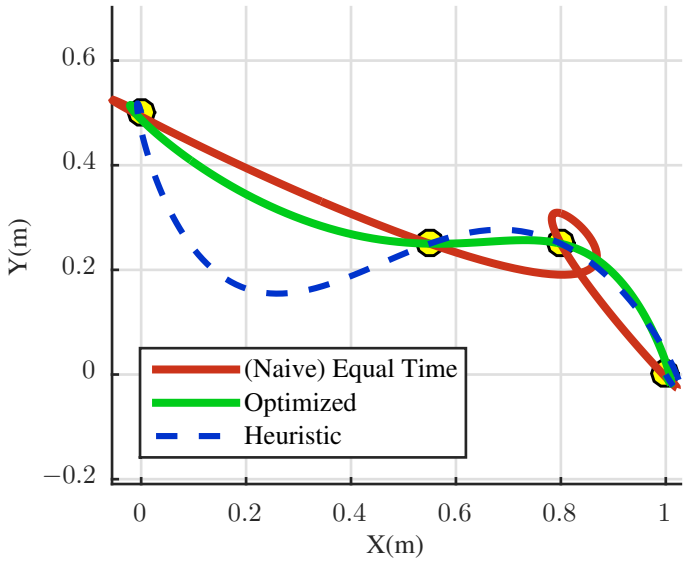


Figure 5.7: Comparison of the optimized trajectories using different methods of time allocation. Red - Equal times (naive), Green - Optimized times, Blue Dashed - Heuristic. This example shows how uneven spacing of waypoints can cause differences between the optimized and heuristic allocation.

There are consequences to using this time allocation method, however, and not all trajectories will look so similar for optimized and heuristic allocated segment times. An example of such a disparity is shown in Fig. 5.3.

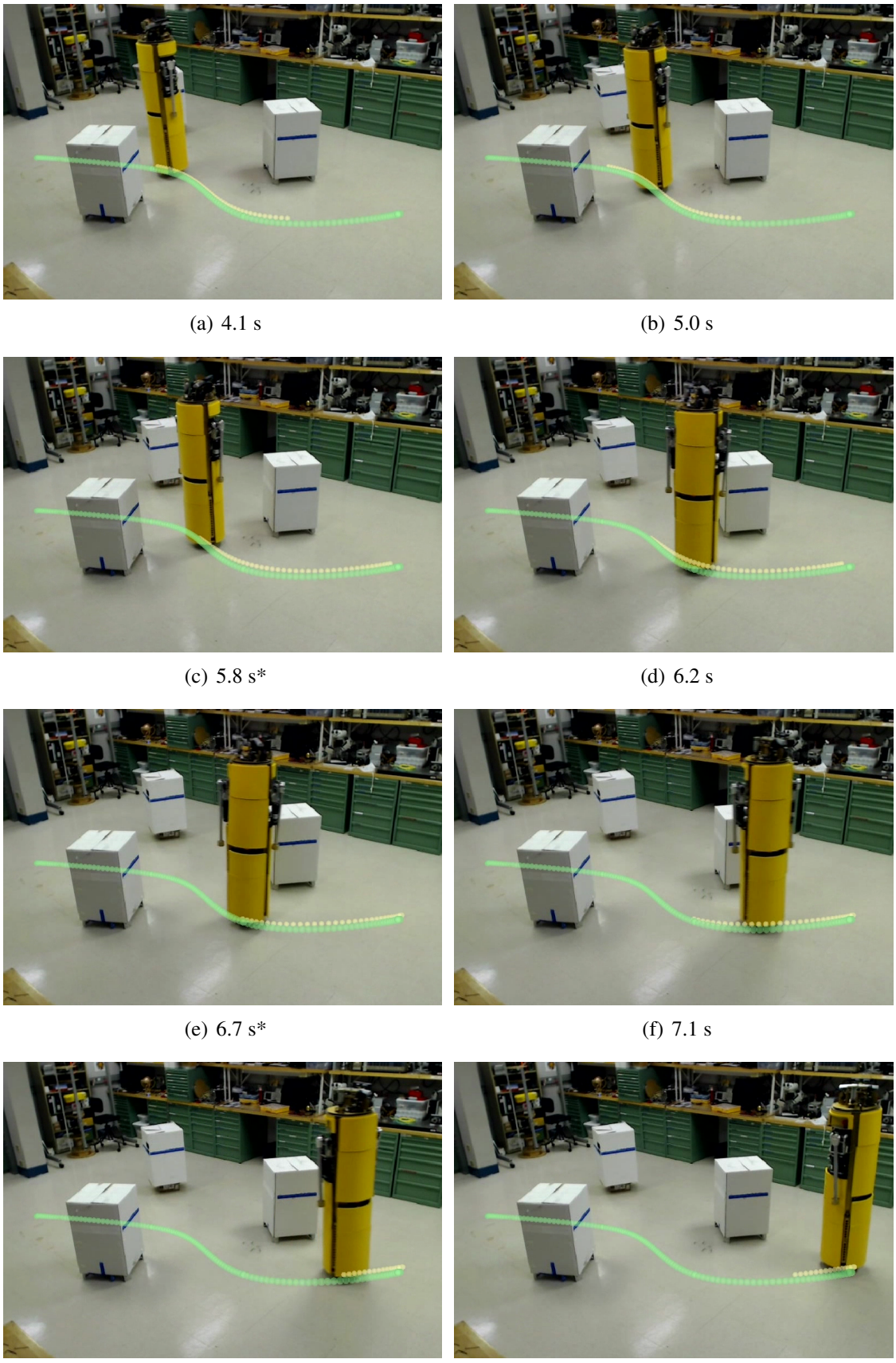
This figure shows that the time-optimized trajectory stays much closer to the waypoints than the heuristically allocated method. The main reason for this discrepancy is the variance in distance between waypoints. The heuristic method works best when the the waypoints are in general regularly spaced. Densely clustered waypoints can cause too much or too little time for individual segments, resulting in larger deviations. These trajectories are certainly still feasible for the ballbot, but are not as desirable. Heuristic allocation is still used in practice on the robot however, because it can operate so much faster than the optimized version. The problems are also mitigated by the fact that waypoints are generated from the graph search paths as described in the previous section. The subsampling of these paths to generate the waypoints means that there are regularly spaced by definition, yielding a good situation to use the heuristic.

5.4 Fast Single-Polynomial Replanning

As discussed in Section 3.4.2, the ballbot uses a 20 m laser scanner and Hector Slam [Kohlbrecher et al., 2011] as a localization solution. This localization information allows for a much better estimate of the robot state than odometry alone. As such, this estimate or a combination of localization and odometry information are usually used for feedback control, as in prior work with the ballbot[Nagarajan et al., 2012a]. Unfortunately, localization systems can yield discontinuous estimates, especially those based on scan matching. The ballbot's system is no different, as these discontinuities can cause jerk in the system, due to discontinous input to the feedback controller.

The method presented in this section takes a different approach than prior work which can utilize localization information immediately while using only continuous odometry for feedback control. Instead of filtering the localization information, a single polynomial trajectory is planned from the current state as estimated from localization to the global trajectory at some time in the future. This trajectory is then executed by a feedback controller using only odometry information, which is always continuous. This trajectory is then continually replanned to take into account updated localization. Generating this intermediate trajectory leverages the ability to compute a single polynomial trajectory in less than a millisecond. Executing feedback control with only odometry information yields another benefit: the planning and control can be decoupled. This is useful in the case of the ballbot because balancing control is done on a real-time computer, while planning and localization are run on a more standard, high-level computer. An example of this method in action can be seen on the next page in Fig. 5.8.

The eights panels of Fig. 5.8 show the ballbot executing a global trajectory, shown in green. The yellow lines show the replanned trajectories over time. The lines are made of dots, discretized by time at 15 dots/second. This give a notion of the planned speed for each replanned trajectory. As stated, these replans help account for discontinous localization updates in a way that still allows for continous, smooth feedback to the controller. This is because the trajectories can be planned exactly from the initial conditions of the robot at any time, since the replanning takes less than a millisecond. These are so fast to generate becuase they are fully constrained polynomials, with boundary conditions of this current known state, and the final conditions of the global trajectory (translated into the local frame via the localization estimate) at time "Lookahead Time" ahead of the current time. This replanning not only helps to accomodate disontinous localization estimates, but also small disturbances and differences from the model. A more detailed look at replanning performance is described in Section 5.5.



(a) 4.1 s (b) 5.0 s

(c) 5.8 s* (d) 6.2 s

(e) 6.7 s* (f) 7.1 s

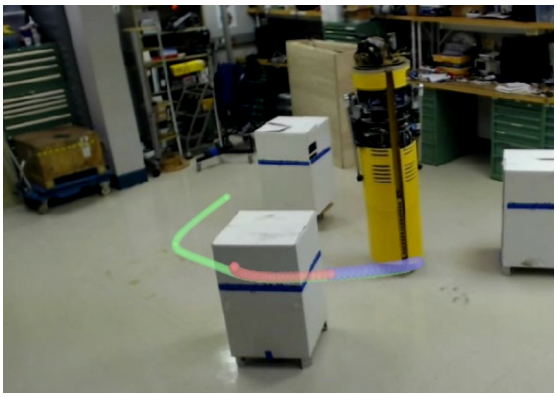
(g) 8.2 s* (h) 9.0 s

Figure 5.8: The ballbot executing a trajectory (green) through obstacles. The yellow paths are the single polynomial trajectories, replanned at a fixed time interval. The snapshots are labeled with the time during the trajectory they are showing. Asterisks indicate a replanning event.

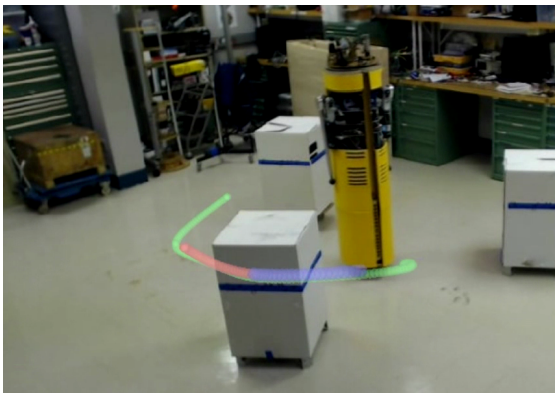
5.4.1 "Cleared" and "Backup" Trajectories

In the case of localization failure or a total high-level system failure, it is desirable for the robot to stay balanced. The previous section introduced the ability to decouple the system in planning and control through the use of replanned, single-polynomial trajectories. This decoupling provides another benefit because if the real-time control system can bring the robot to rest by itself in the event of a high-level failure, the robot is much safer. To this end, the proposed trajectory generation method was designed to only ever send 1.2 s of the desired trajectory to the real time system, followed by another trajectory which brings the system to rest, as in the recovery trajectories shown in Chapter 4. As long as everything works properly, the high-level replanner sends another 1.2 s trajectory before the previously sent trajectory has been completed. If the high-level planning fails for any reason or generates a plan that is infeasible, the low-level system reverts to the backup trajectory and comes to rest. This value of 1.2 s was chosen empirically trading off safety and computational load. An example of this backup trajectory method can be seen in Fig. 5.9.

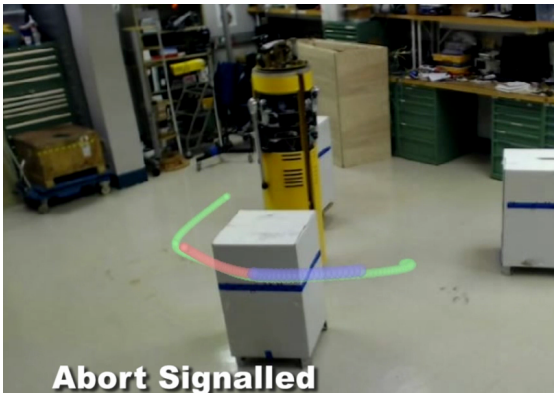
The figure shows execution of a green, global trajectory through boxes, similar to Fig. 5.8. Also like the previous example, the replanner is generating new trajectories at a fixed frequency. In this case, the period of replanning is 1.1 seconds, such that there will always be a new trajectory before the 1.2 second "Cleared" trajectory is completed. The yellow replanned motion is not show in Fig. 5.9; instead, the the blue lines show the first 1.2 seconds of the replanned polynomial trajectory. The red lines that follow are another generated polynomial trajectory that would bring the system to rest. These are unconstrained in their final position, and optimized to have minimum energy, just as in the recovery motion of Chapter 4. The figures shows what happens when a high-level system failure is simulated. Because the controller received no new trajectory after completing a "Cleared" motion, the system reverts to the stopping motion and comes to rest. This means the ballbot always has a safe, dynamically feasible backup plan, that brings the system to rest. If this system did not exist, the robot would either continue executing the last commanded lean angle or discontinuously switch to a station keeping controller. Neither of these are good options. As such, this method results in a hierarchical system with multiple levels of safety consideration.



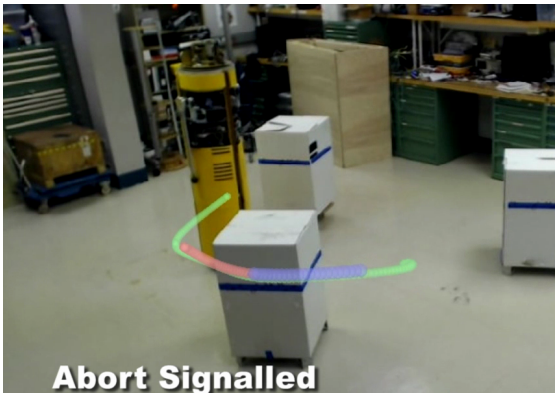
(a) The ballbot beginning exectution of a new global trajectory (green).



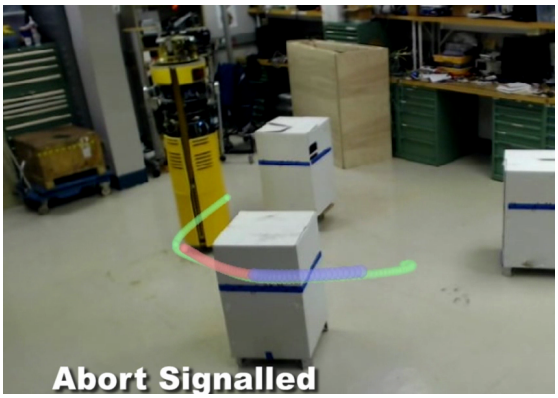
(b) A replan event generates a new trajectory. The blue line indicates the first 1.2 seconds of that trajectory. The line shows a trajectory that would bring the system to rest.



(c) An abort is signalled, meaning no new trajectory will be replanned, and the robot will default to using the red stopping trajectory.



(d) After completing the 1.2 second blue motion, the ballbot executes the red stopping trajectory.



(e) The robot completes the red trajectory, coming to rest.

Figure 5.9: Example of "Cleared" and "Backup" Trajectories with the ballbot.

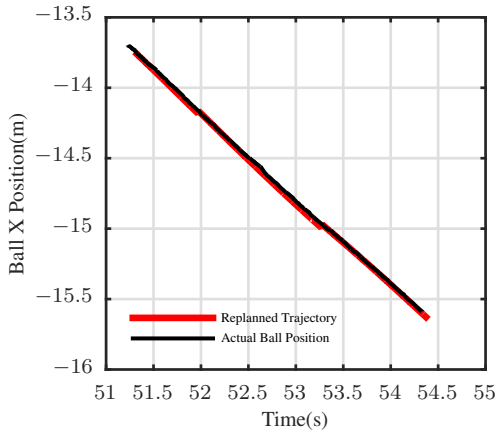
5.5 Experimental Validation

Trajectories were tested extensively over 60 trials, in a 5 m $\times$ 8 m room in the presence of static and dynamic obstacles. Average velocities were varied from .3 m/s to .7 m/s. Ten experiments tested the stability of generating long trajectories. These trajectories were all over 20 m and traversed rooms and hallways with floors of carpet and tile. The timing of the trajectory segments were allocated using the heuristic method discussed in Section 5.3. Plans were checked once per second for continued feasibility. If an obstacle had invalidated the planned trajectory, a new trajectory was generated. New goals were also commanded to the robot while already executing a trajectory to assess the ability to smoothly transition to a new trajectory for a new goal.

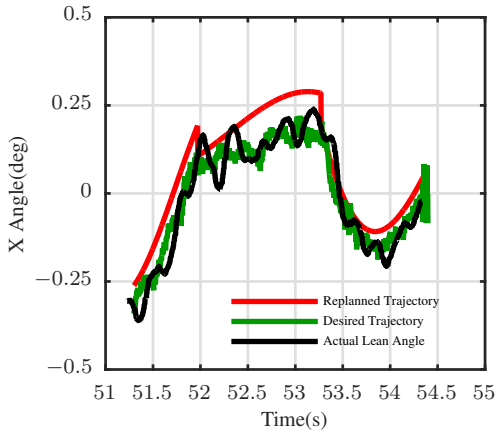
The ballbot was able to successfully navigate a building in the presence of dynamic obstacles at speeds up to .7 m/s. Fully dynamic, feasible trajectories up to 25 m long were planned in less than 50 ms. A piece of one such trajectory is shown in Fig. 5.10, with the full trajectory shown in Fig. 5.5. This particular trajectory was 90 s long, moving through 2 doorways and a hallway at .6 m/s. Fig. 5.10 highlights the replanning for localization, as well as the tracking accuracy of the system.

Fig. 5.10(a) shows the ball position tracking of the trajectory. As this particular segment of the trajectory was mostly in the x direction, over the 3 seconds shown, the ballbot moved almost 2 m. As such, the difference between the desired trajectory and the actual performance is almost indistinguishable at this scale. Shown in Fig. 5.10(b) is lean angle in the x direction. Replanning to account for localization occurred at t = 52 s and 53.2 s. Fig. 5.10(c) shows the tracking in the y direction. Note the scale, as this figure shows only .25 m of travel. This figure clearly shows that the replanning events use the state of the ballbot, both lean angle and position, as the initial conditions.

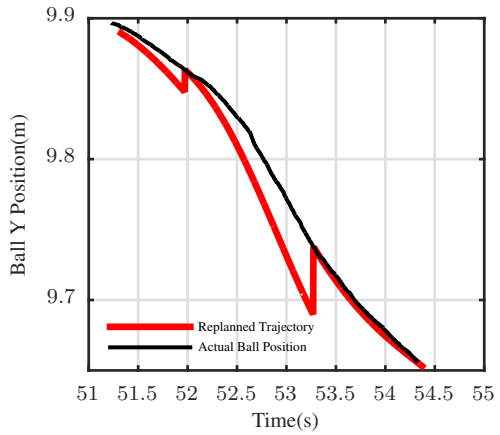
The replanning event at t = 52 s actually uses a poor localization estimate, off by .05 m. As such, it looks like the ballbot is .05 m from where it should be in the y direction, but cannot compensate by the time the next trajectory is planned. By the time the next replanned trajectory is generated at t = 53.2, the localization has returned to a better estimate, as is clearly seen from the smoothness and good tracking of the next replanned segment.



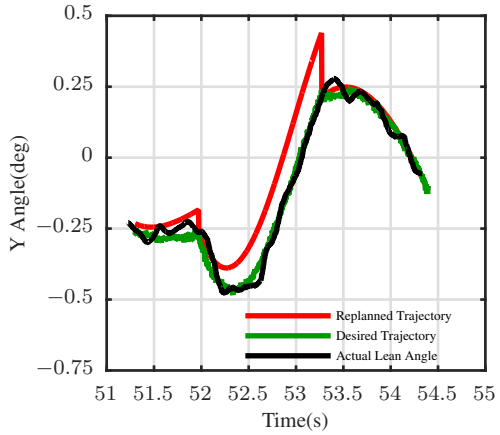
(a) Ball x Position tracking over 2 m in 3 s



(b) x Lean Angle tracking. The planned trajectory is in red, the controller output in green, and the actual angle in black



(c) x Position tracking .25 ms in 3 s. This portion of the total path was primarily moving in the x direction.

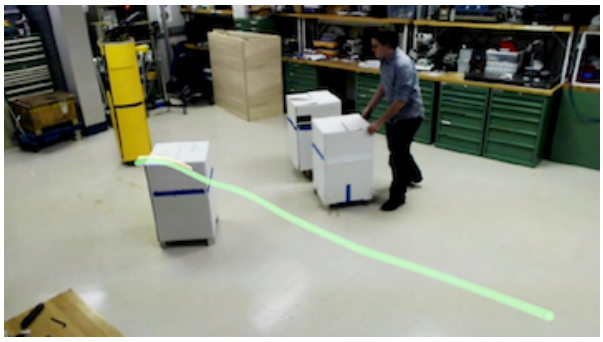


(d) y Lean Angle tracking. The planned trajectory is in red, the controller output in green, and the actual angle in black

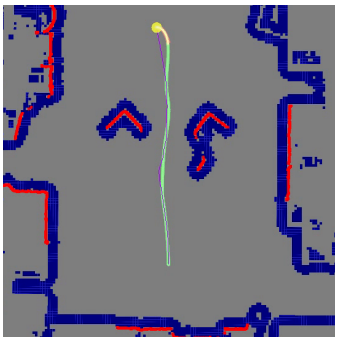
Figure 5.10: Planned and actual state data from an experiment with the ballbot. The entire experiment had a duration of 90 s and traversed over 20 m through 2 rooms and a hallway with obstacles. Seconds 51 through 55 are shown to highlight position and lean angle tracking along with 1.2 s period replanning strategy. The replanning that occurs at t = 52 s has relatively poor localization, but the replannned trajectory at 53.2 s returns to the global trajectory very adequately.

5.5.1 Dynamic Obstacles

Because this method of trajectory generation is so fast generating plans in less than 50 ms, it is very appropriate for dynamically replanning in the presence of moving obstacles. This is achieved quite simply, as the planned trajectory is continually reevaluated in the costmap at a fixed frequency. If the trajectory becomes invalidated by an obstacle, a new plan is generated using the current state of the robot. This capability is shown in Fig. 5.5.1. Fig. 5.11(a) shows the initial situation, where the ballbot generates a trajectory shown in green. At 4 s into the experiment, a box is placed directly in the path of the robot, and the robot replans a trajectory to its right, as seen in Fig. 5.11(c). It is important to note that all planned trajectories begin with the ballbot's current state as the initial condition. As such, the robot smoothly transitions from an old trajectory to a new one. Lastly, the second path of the ballbot is blocked by a person, as shown in Fig. 5.11(e).

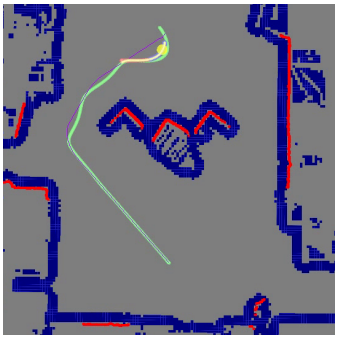


(a) The ballbot initially starting at rest at $t_0 = 0$ s. A straight path to the goal in the lower right is planned.

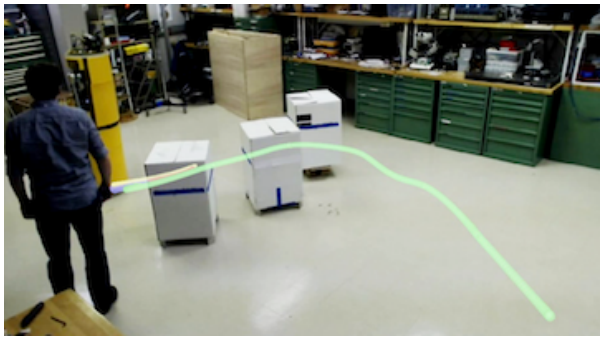


(b) Overhead view of (a)

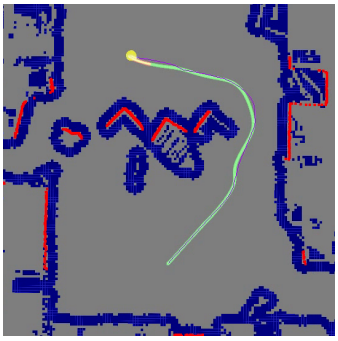
(c) At $t_1 = 4$ s, a box is placed blocking the straight line path, and the ballbot replans smoothly to go around the obstacles.



(d) Overhead view of (c)



(e) At $t_2 = 11$ s, a person block the path of the ballbot again



(f) Overhead view of (e)

Figure 5.11: The ballbot navigates a room in the presence of dynamic obstacles. The planned trajectory is shown in green; laser scan, red; Inflated obstacles, blue, and the current ball position, yellow

5.6 Summary

This chapter presented a method for generating dynamically feasible trajectories for the ballbot in the presence of obstacles. This method has been shown to produce smooth motions, even over very large distances. The generation of these paths is also sufficiently fast for replanning when faced with moving obstacles, and replanning to account for localization information. Trajectories produced have been experimentally verified on the ballbot at speeds up to .7 m/s. This is a significant speed for a person sized robot, matched by very few other systems. The method presented also always ensures both a smooth trajectory and smooth transitions between segments. Furthermore, to our knowledge, this is first time a method has successfully enabled a ballbot to navigate unstructured environments at such speeds or distances. This advancement provides a real, tractable solution to autonomous ballbot indoor navigation.

Chapter 6

Physical Human-Robot Interaction with the Ballbot

This chapter introduces three cases of physical human-robot interaction with the ballbot. The first is physical assistance in helping people out of chairs. Next, the problem of leading people by the hand is explored. Lastly, a possible strategy for navigating through dense crowds of people is presented. Investigating these aspects of pHRI included three human subject trials and one technical feasibilty study.

6.1 Proxemics in Ballbot Planning

In a step toward physical interaction people, a user study was run in 2013 to assess the unique requirements in approaching people during a collaborative task [Shomin et al., 2014]. Specifically investigated was whether curved trajectories that respected proxemic zones of interaction were qualitatively better than straight trajectories that violated a user's personal space.

Subjects' body language, reported comfort levels, and behavior were observed in videos made during a collaborative task. Video analysis and qualitative comments made during and after the collaboration indicated that while participants rated the straight trajectories the highest, they were comfortable with curved approach trajectories (none of which entered the subject's personal zone) and with the experience of interacting with the robot overall. A representative approach trajectory can be seen in Fig. 6.1.

The effect of varying speeds of approach was also studied. The maximum speed of .6 m/s was described as too slow for the collaborative task. While the difference was noticed, there was no discomfort with peak speeds or variance in speed. Additionally, the findings of [Mutlu et al., 2006] were confirmed, in that subjects overall became more comfortable with the robot over the course of the experiment.

The study was a successful step in our program of pursuing physical HRI research with the ballbot. Furthermore, all participants reported having no fear of the robot, which we consider to be successful in the safety and control of the system [Shomin et al., 2014].

Figure 6.1: The curved trajectories planned and executed by the ballbot respect intimate and personal zones of the subject, whereas the straight line paths do not necessarily do so. Proximal zones are shown around subject: Intimate - Orange, Personal - Green, Social - Red

6.2 Physical Assistance in Sit-to-Stand

Many people need assistance in standing which is both a nuisance and often a danger to the people who offer help. This section presents a first step toward easing the burden on health care professionals by employing a dynamically balancing robot, the ballbot, to help people get out of chairs.

Assisting people in getting out of chairs, or more formally: sit-to-stand (STS) assistance has had relatively little relevant literature. However, STS as an evaluative and rehabilitative measure has been studied for many years. Kerr et al. studied individuals sitting and standing, observing time of completion and separating the sitting motion into different phases: forward lean, knee extension, vertical displacement, and recovery [Kerr et al., 2003]. This study noted that the elderly were slower in both sitting and standing. Sibella et al. found that obese individuals use less trunk flexion and consistently shift their feet back under the chair causing a lower hip torque but a high knee torque [Sibella et al., 2003]. One study that actually looked at assistive technology in egress from chairs used an ejector seat [Munro et al., 1998]. It was found to relieve the hip torque demand for elderly people. This study also, however, highlighted that taller chairs are inherently easier to get out of, and may be preferred over an ejector.

Very little research has been conducted with human-scale robots assisting people physically with large forces. One recent exception is a study of human-robot cooperative manipulation [Mortl et al., 2012]. In this case, it was actually preferable to keep forces between the robot and participant to a minimum, as high forces suggest disagreement. This study did measure forces up to 40 N, however, making it a physical Human-Robot Interaction (pHRI) study of human proportions with an assistive robot. Another robotic study with relevance to the topic of assisting seated individuals is the work [Dautenhahn et al., 2006] of Dautenhahn et al. This study, entitled, "How may I serve you?" investigated how best a robot should approach a seated person in an assistive scenario.

6.2.1 Human-Human Sit-to-Stand

Subject Trial

In order to gain insight for a robot assisted STS control strategy, a subject trial was conducted to observe how people help other people out of chairs [Shomin et al., 2015]. Fifteen subjects were recruited through online advertising to participate. Nine females and six males participated in the study. All were able bodied, ranging in ages from 18 to 63. The group's average self-reported familiarity with robots on a 1-7 scale was 3.07.

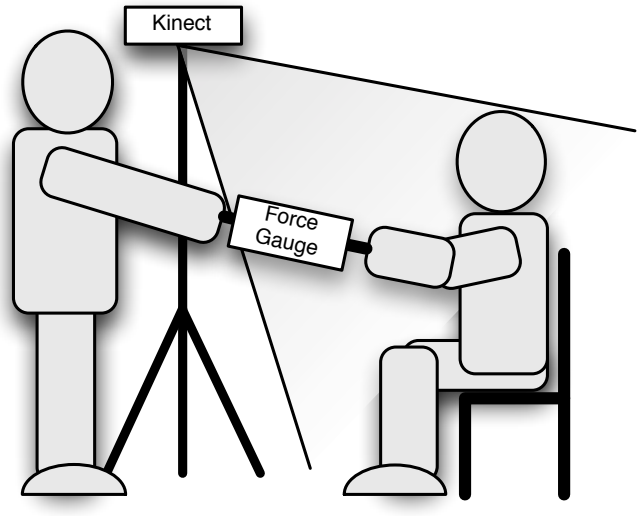
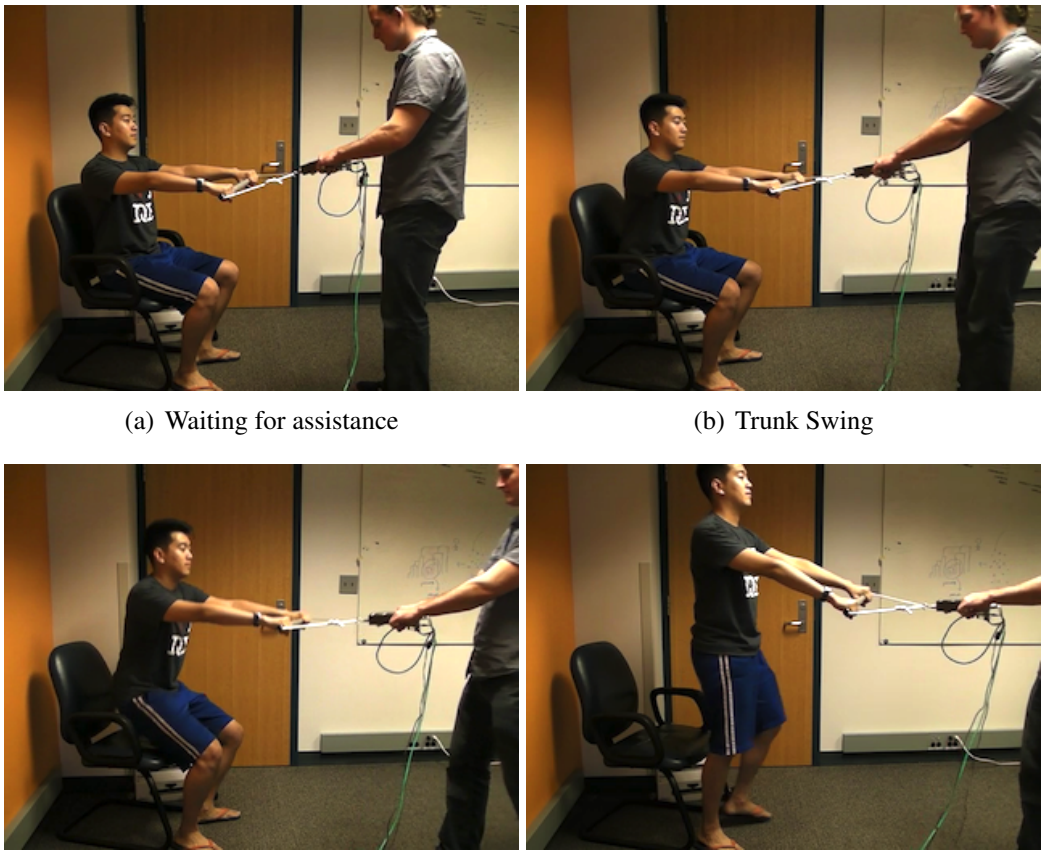


Figure 6.2: Participants of the subject trial were assisted in rising out of a chair while their pose was tracked by a Microsoft Kinect. A force gauge measured how much force an experimenter provided in assistance.

Participants were asked to sit comfortably in a chair. The experimenter, standing, offered the subject a bar attached to a force gauge which the subject could take with both hands. The experimenter then assisted the subject in rising from the chair, by pulling on the other side of the force gauge. During this operation, the subject would be tracked by a Microsoft Kinect running OpenNI [Interaction, 2010] skeletal tracking software. A diagram of this experimental setup can be seen in Fig. 6.2.

Subjects were assisted in rising from four different chairs, four times each. They were free to place their feet wherever they preferred. Subjects were also assisted without the force gauge and given a questionnaire, however, these qualitative components of the study could be used for future work.

A subject can been seen participated in the experiment in Fig. 6.3. Fig. 6.3(a) shows the participant holding the bar attached to the force gauge waiting to start getting out of the chair.



(a) Waiting for assistance (b) Trunk Swing

(c) Leg Extension (d) Stable Equilibrium

Figure 6.3: A subject is assisted by the experimenter out of the chair using a force gauge while being tracked by a Kinect.

In Fig. 6.3(b), the subject begins to swing his trunk and then proceeds to extend his legs in Fig. 6.3(c). The subject is at a standing equilibrium in Fig. 6.3(d).

Force and Skeletal Data

Data from the subject trial was parsed into joint trajectories and corresponding force curves for each STS experiment. One such experiment's data can be seen in Fig. 6.4. The skeleton of the subject can be seen on the left with four colors: green, red, cyan, and purple showing the stages of getting out of the chair. The force curve on the right has four colored dots which correlate with the skeleton at each particular time.

The joint trajectories were examined in an attempt to find the most consistent set of joints to use as a cue for the STS motion. The shoulder trajectory was found to be the most repeatable and intuitively shows a strong cue as to where in the STS motion a participant is. This follows naturally, as the shoulder trajectory can show both trunk swing and the rise in elevation from leg extension. One particularly consistent subject's data is shown in Fig. 6.5. The red lines show 11 trajectories of shoulder X and Z position along with the assistive pulling force. The blue lines

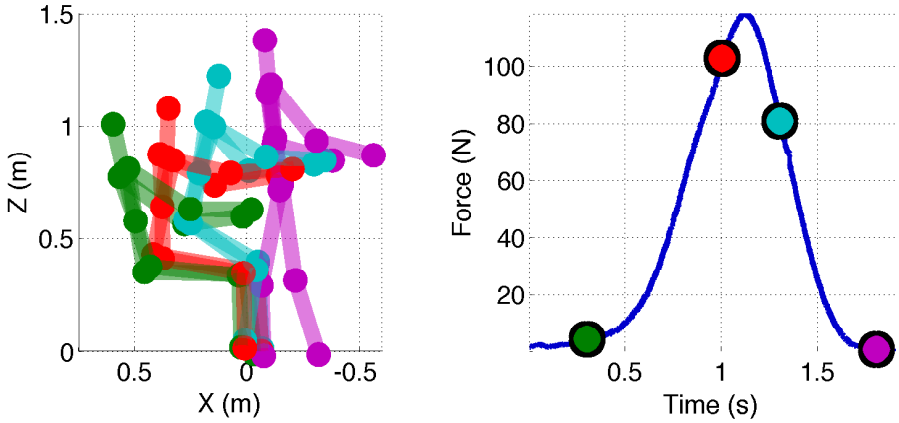


Figure 6.4: Side view skeletal tracking data and force data for a single experiment. The color of the skeleton corresponds to the color dot in the force plot.

show the average of all 11 trials, scaled in time to match starts and ends.

6.2.2 Model and Control

Ballbot

For the purpose of assisting people in STS, a planar model of the system with arms is used, as seen in Fig. 6.6. A planar model is appropriate as STS is a planar problem, symmetric about the sagittal plane. The ballbot has an independent yaw drive, which is assumed to account for any out of plane deviations.

Since the ballbot runs an inner-loop attitude controller, the problem of creating a desired force becomes one of finding the appropriate lean angle. To find this angle, a quasi-static assumption is enforced. Although STS is a dynamic maneuver, the largest component to the assistive force by far is the lean angle. Also, without this assumption, there is not a one-to-one analytic correspondence. With the quasi-static assumption, simply solving $\sum F_z = 0$ yields

$$F_{exp} = \frac{(M_b + M_w)gl\sin(-\phi)}{r + (d\cos(\phi)\cos(\frac{\pi}{2} - \psi))}. \tag{6.1}$$

F_{exp} is the expected tensile force along the arm for a given lean angle, ϕ, and arm angle, ψ. The goal, however, is to find the appropriate lean angle for a force. Solving (6.1) for ϕ yields

$$\phi_{cmd} = \mathrm{a}tan2(\frac{rc + \sqrt{b^4 - b^2r^2 + c^2b^2}}{b^2 + c^2}, \frac{-r}{b} + \frac{c(rc + \sqrt{b^4 - b^2r^2 + c^2b^2})}{b(b^2 + c^2)}) \tag{6.2}$$

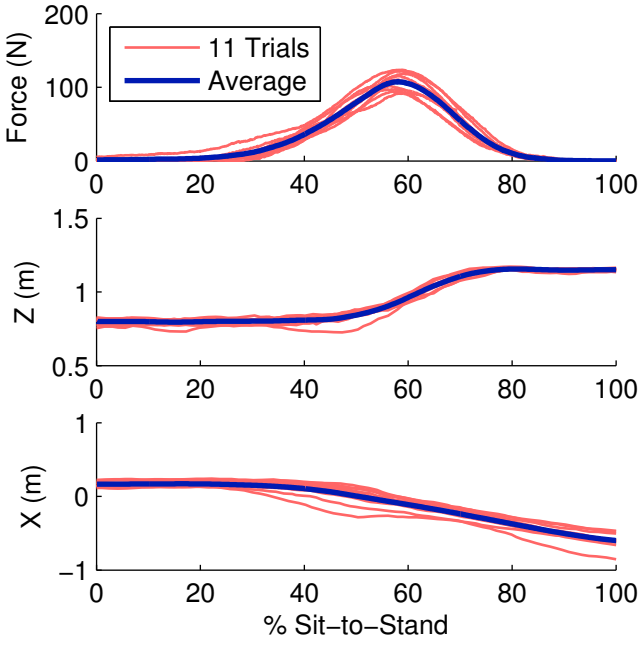


Figure 6.5: Shoulder x and z coordinates for a single participant across 11 trials. The force and position for each trial are shown in red with the average shown in blue. The times have been scaled to match the starts and ends of each trial.

where

$$b = d\cos(\frac{\pi}{2} - \psi), \tag{6.3}$$

$$c = \frac{(M_b + M_w)gl}{F_{\text{des}}}. \tag{6.4}$$

There are actually two solutions for ϕ, but since F is constrained to be a positive tensile force, this is the only valid solution. (6.2) give the ability to replay force trajectories with the robot, as will be seen in Sec. 6.2.3.

Impedance-based Controller

Replaying force trajectories is not a tractable solution to the STS assistance problem. A human participant could easily miss the opportunity while the robot is pulling. As such, it is beneficial to design a control law that can reproduce the force trajectories from the subject trials, but using a cue that does not depend on time. As previously discussed, the shoulder position is a very repeatable cue to the state of the STS cycle.

A simple impedance controller is chosen which acts simply as a spring-damper system

$$F = k(s_{\text{eq}} - s_{xz}) - b\dot{s}_{xz} \tag{6.5}$$

where s_{xz} is the shoulder position in the X-Z plane, s_{eq} is the equilibrium position of the shoulder where no force is applied, and $\dot{s}_{xz}$ is the velocity of the shoulder. Motivation for such a control

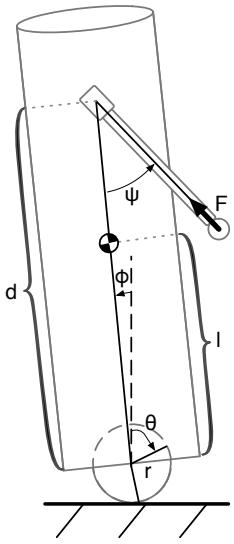


Figure 6.6: The planar ballbot model with arms. State variables are lean angle ϕ, ball angle θ, and arm angle ψ. Distance from the center of the ball to the shoulder joint is d. Distance from the center of the ball to the center of mass (COM) is l, and the ball radius is r. F is the assistive force, in line with the arm. Not pictured, the mass of the ball (wheel) is M_w, and mass of the body M_b.

law comes from active prosthetics [Sup et al., 2008]. Sup et al. use piecewise impedance functions in different phases of the gait to control a transfemoral prosthetic. These control laws are appealing as they are simple for the user to comprehend and individually stable. For the STS maneuver, two control phases are chosen. This is because of the two natural equilibria at the start and end of the STS. As such, k, b, and s_{eq} in (6.5) are set by

$$k, b, s_{eq} = \begin{cases} k_1, b_1, s_{eq1} & \text{if } s_z < z_{thresh} \\ k_2, b_2, s_{eq2} & \text{if } s_z > z_{thresh}. \end{cases} \tag{6.6}$$

To find the optimal k_1,b_1,k_2,b_2, s_{eq1}, and s_{eq2}, the sum squared difference between the average force trajectory from the human subject trials and the output of (6.5) is minimized:

$$\min \quad \sum_{i}^{n} (F_{\text{des}} - F)^2. \tag{6.7}$$

This is accomplished via an SQP solver in MATLAB. The optimal k's and b's are $k_1 = 392.1$ N/m, $b_1 = 0.0$ Ns/m, $k_2 = 324.3$ N/m, $b_2 = 16.77$ Ns/m. The optimal equilibrium positions are shown in Fig. 6.7 in black, plotted on top of the average shoulder trajectory. The equilibrium points lines up very nicely with the start and end of the shoulder trajectory. This is to be expected as there is no assistive force required at the start or end of the STS. The force profile from the optimized values can be seen in Fig. 6.8. Because the output is discontinuous at the

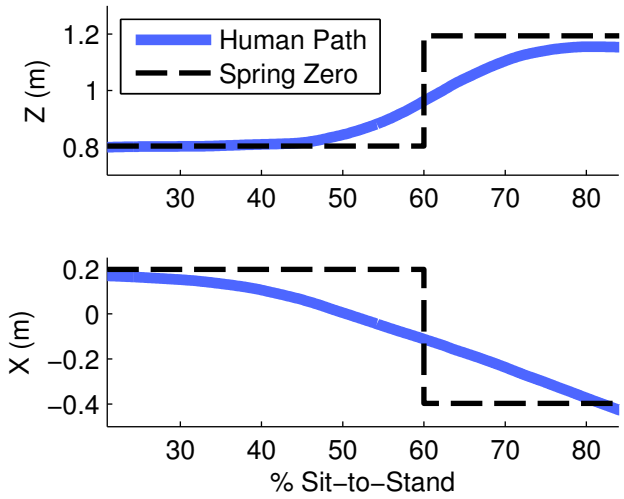


Figure 6.7: Shoulder trajectory (blue) in x and z as shown in Fig. 6.5. The dashed black line shows the equilibrium position of the impedance controllers, optimized to fit the data. Unsurprisingly, the equilibria are very close to the initial and final positions of the subject's shoulder.

switching point, a low-pass filtered signal is also plotted with a time constant of 50 ms. This is an empirically chosen value to avoid exciting any higher order modes in the system.

Notice that the first phase of the controller which is active during the trunk swing and beginning of leg extension is not actually acting like a "spring." In fact, it behaves as an anti-spring, with s_{eq1} as an unstable equilibrium. This is a necessary behavior as the first half of this assistance must input energy to the system. Furthermore, it must increase in force as the body moves. This is also not a problem as the robot will only apply tensile forces, so it cannot push the participant down into the chair past the s_{eq1}.

6.2.3 Experimental Validation

Three sets of experiments were carried out to assess the validity and feasibility of the proposed STS assistance method. Lean angle trajectories were replayed to assess the ability of the system to track large lean angles. Next, force trajectories were replayed on the system, taking into account arm angle. Lastly, the full controller was tested. An experimenter stood in as the STS participant.

To ensure safety, the robot was tethered with a slack line to an aluminum frame overhead. This ensured that if the ballbot went unstable, it would not fall on any experimenters. The tests were also done with another researcher near the emergency stop on the robot, which disables power to the ball motors and the arms.

The lean angle tracking from the first set of experiments is shown in Fig. 6.9. The RMS error is $.18^\circ$ with a maximum deviation of $.7^\circ$ at the max lean angle of 12.4°.

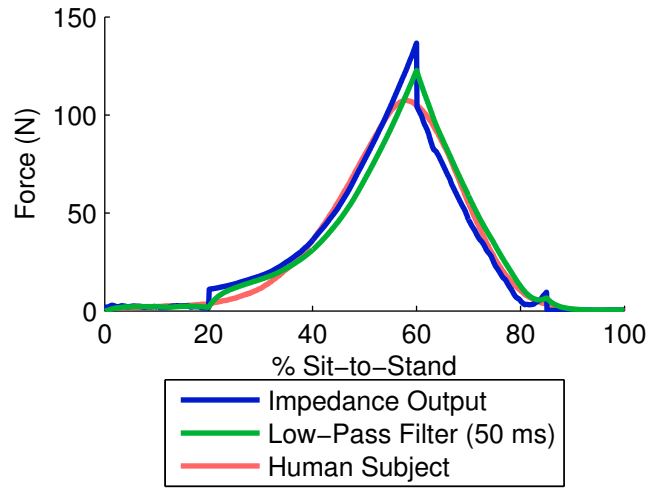


Figure 6.8: Average force profile from human subject (orange) shown against the output of the impedance based controller (blue). The output of the controller is discontinous and as such is filtered to ensure smooth operation. A first order low pass filter with 50 ms time constant is shown in green.

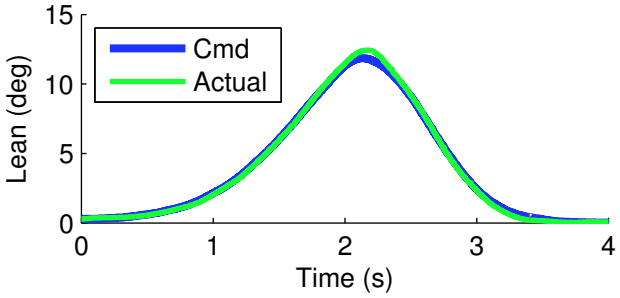


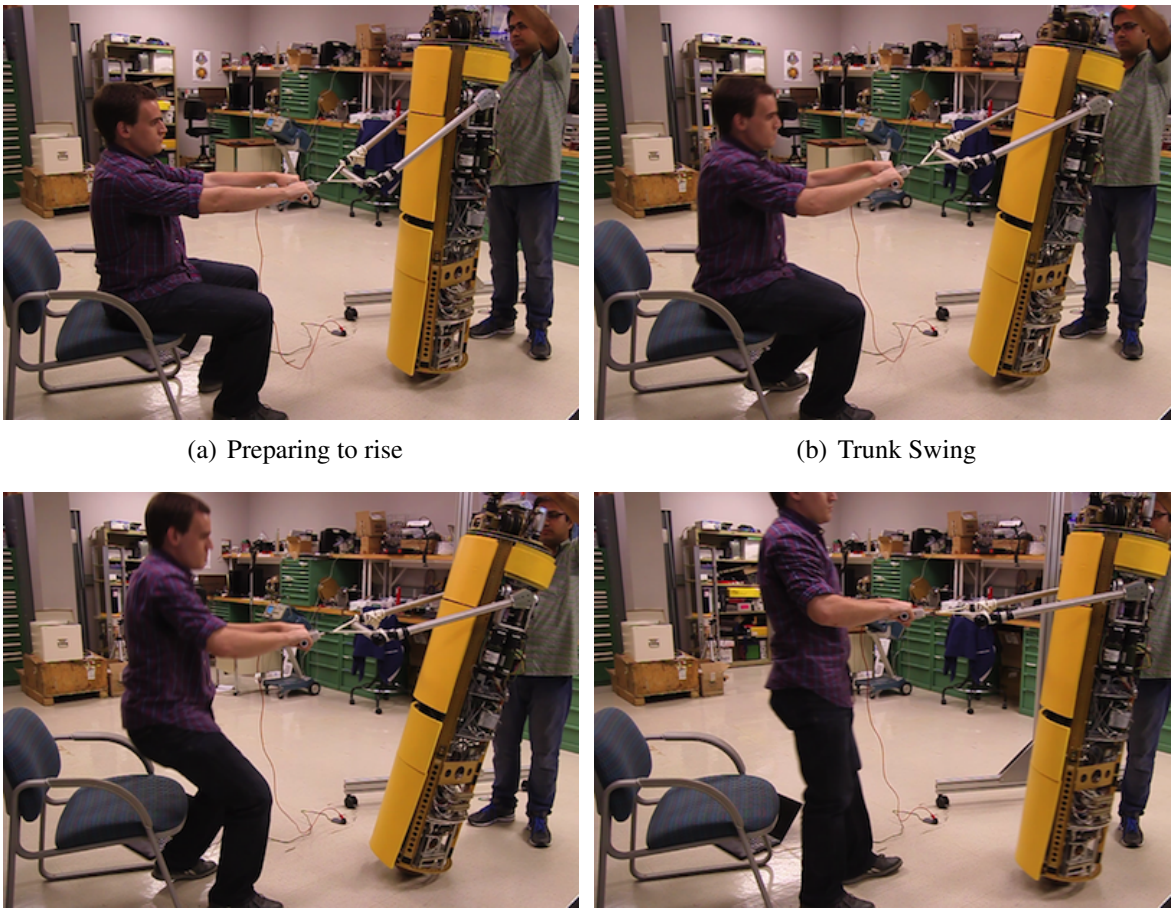
Figure 6.9: Desired and actual lean angles from a replayed force trajectory

Force Trajectory Replay

A replay of the average human subject force trajectory on the ballbot is shown in Fig. 6.11. This experiment attempted to create the same force profile as the average human subject trial, taking into account the arm angle, ψ, dynamically. Note that there is no direct feedback on this force, as there is on lean angle. Still, the tracking is acceptable and stable with the RMS error of 16.2 N, however the maximum deviation was 38.5 N.

Impedance Controller

To run the impedance-based controller, the shoulder position of the subject must be known to update the control law. To avoid the risk of poor quality live skeletal tracking, it was assumed that the subject's hands were holding the ballbot's arms for the duration of the experiment and that their relative geometry did not change.



(a) Preparing to rise (b) Trunk Swing

(c) Leg Extension (d) Standing Equilibrium

Figure 6.10: The ballbot assisted a person in standing using the impedance based controller. In (c), the robot is applying over 100 N of assistive force. An experimenter stands behind the robot in the event of a failure to press the emergency stop.

The low pass filter shown in Fig. 6.8 was used to smooth discontinuities. Furthermore, b_2 was empirically lowered to from 16.77 to 5 Ns/m. Lastly a debounce timer of 300 ms was put into the switching law prohibiting the system from switching states more frequently than every 300 ms. This was done to avoid instability caused by nuisance switching at the boundary of the two controller regions.

The result of this experiment is shown in both Fig. 6.10 and Fig. 6.12. Fig. 6.10 shows the participant being assisted by the ballbot throughout the course of an STS.

Fig. 6.12 shows the force applied to the participant over the course of experiment. Note that the profile is markedly distinct from Fig. 6.5 because the participant chose to get up more slowly, taking 12 seconds to complete the STS. This shows one of the main advantages of this method. The assistive force is dependent on the position of the participant, which they have control over.

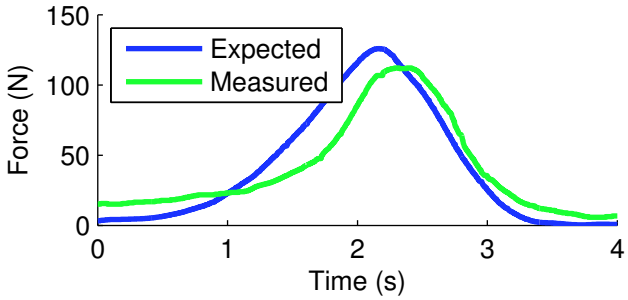


Figure 6.11: Desired and actual force from a replayed force trajectory. This takes the arm angle ψ into account from (6.2)

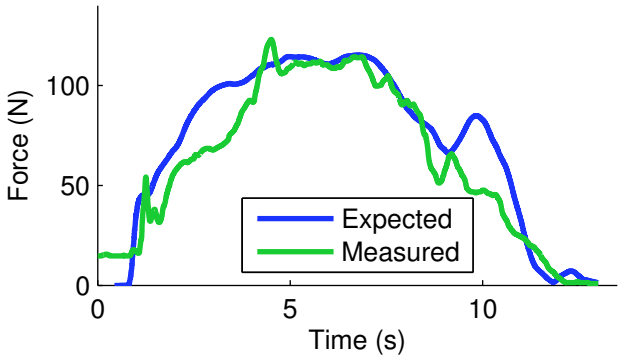


Figure 6.12: Desired and actual force using the impedance based controller. These desired and actual forces correspond to the experiment shown in Fig. 6.10

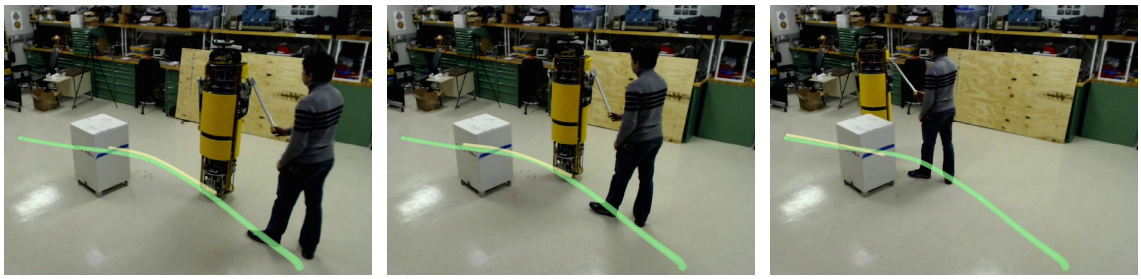
6.2.4 Summary

Using force and joint data gathered through human subject trial, an impedance-based controller was designed and experimentally validated. The feasibility of this control scheme was demonstrated on the ballbot assisting a person with over 120 N of force. This is a step toward robotic technology influencing people's daily lives for the better.

6.3 Leading a Person by the Hand

For many visually impaired and elderly people, public and unfamiliar places remains a challenge, and better navigation through physical assistance can improve their quality of life. Navigational capabilities can already be augmented through the use of seeing-eye dogs and walkers, respectively. This section present a means of robotic assisted navigation.

To achieve the goal of assistive navigation, we propose to first use the planner described in Section 5 with some modification. Firstly, since the planner is based on the ROS navigation stack, it can be efficiently modified to accommodate a larger obstacle inflation associated with the joint footprint of the ballbot and person. The previously studied planning infrastructure produces obstacle-free target trajectories to a destination without external forces. We hypothesize reasonable and useful leading could be achieved by continually replanning a single polynomial trajectory to the global target path. This is an extension to the constant replanning shown in Section 5.4 As the ballbot executes a planned global trajectory, its linear tracking controller on lean angle induces a force at the hand, just as in the case of assisting people out of chairs. This strategy is essentially free of a model of the person being assisted, which is beneficial in its simplicity. An early test of this system's feasibility can be seen in Fig. 6.13.



(a) (b) (c)

Figure 6.13: The ballbot physically leading a person along the green trajectory around an obstacle.

In the figure, the green path is the global trajectory around an obstacle to the goal. The yellow path is the replanned single polynomial, in this case a three second feasible trajectory being replanned every .5 seconds. Unlike the replanning for localization in prior work, the goal of the yellow replanned trajectory is not time parametrized. Instead, at each replanning event, the closest point in the global trajectory is found and the feasible trajectory is planned to (in this case) three seconds ahead of that point in the global path.

6.3.1 Replanning and State Assessment

The replanning algorithm presented in Section 5.4 must be modified to accommodate leading by the hand. Namely, as presented, the end condition of the replanned polynomial is time parametrized and marches along the global trajectory. It is always planned to be "Lookahead Time" ahead of the current desired state in the global trajectory. This is incompatible with leading as the person being led may not be able to keep up with the robot. If the person chooses to

stop or go slower, a different strategy is necessary. As such, instead of time parametrizing the replanned polynomial, the current state is assessed and compared to the global plan. Namely, the global trajectory is sampled at a discrete time step and the L2 norm of the current state is calculated with respect to this sampling.

Replanning Period	0.3 s
Polynomial Lookahead Time	4.65 s
Maximum Planned Lean Angle	3.78°
Maximum Planned Velocity	1.05 m/s
Polynomial Sampling Timestep	0.01 s
State Threshold	0.08

Table 6.1: Replanner Parameters for Leading

Table 6.1 shows the empirically tuned parameters for such a replanning strategy for leading by the hand. Note that a "State Threshold" is chosen such that if the norm of the state to the current time parametrized point in the global trajectory is less than this threshold, it will be used instead of the closest sampled point. The key to this method is the choice of parameters for the replanner. The most important parameters are the maximum planned lean angle, the frequency of replanning, and the lookahead time. From the maximum lean angle, the maximum force on the person can be calculated from Eq. 6.1. The frequency must be sufficiently high to reduce oscillations. A low frequency yields oscillations because the steady state of the system while moving is zero lean angle at constant velocity.

The advantage of this system is that no force will be exerted if the person is moving at the same speed and in the same direction as the ballbot. Deviations from the path, both spatially and temporally, will result in force based on the PD tracking controller and the lean angle to force mapping. Therefore, if the robot starts moving and the person remains stationary, the robot saturates its maximum allowed lean angle, yielding the maximum planned force. Once the person starts moving, a steady state lean angle will no longer be required and the force decreases.

6.3.2 Experimental Setup

In order to validate the ability to lead people by the hand, a user study was performed with the ballbot. This study enrolled able-bodied participants with the intention of leading subjects between goals in the Microdynamic Sytems Lab. Fig. 6.14 shows the locations of six goal locations denoted by letter in the lab.

The experiments were designed for the robot to lead subject between the six goals. These trials included runs in which the subject knew the goal to which they were being led, in addition to runs where the goal location was unknown beforehand. More details are explained in Section 6.3.3.

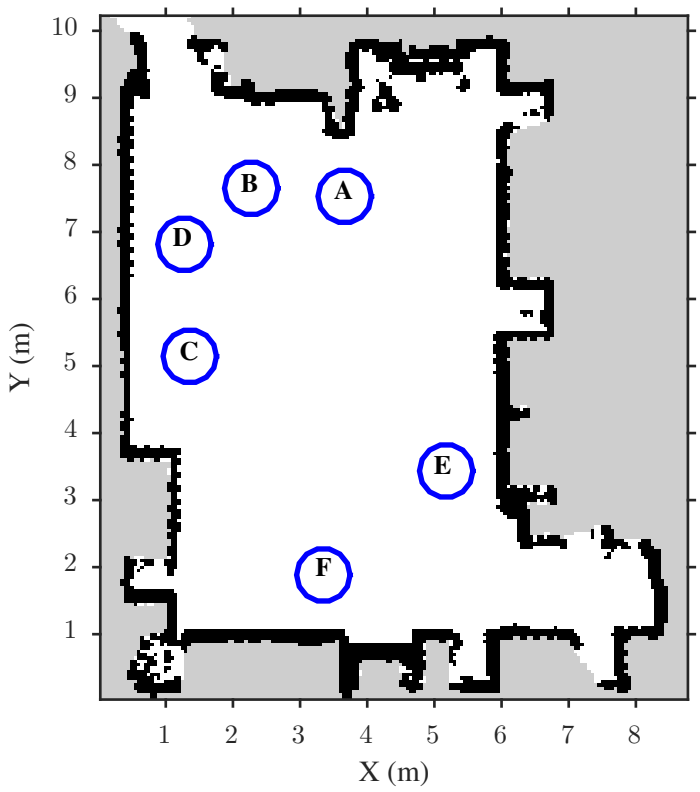


Figure 6.14: Lab setup and goal locations for human subject trial on leading by the hand.

6.3.3 Hypotheses

The parameters for replanning and control in these experiments were chosen to bound interaction for (F_{exp}) to 25 N. This is not a given, however, as this assumes that the subject will not jerk the arm significantly or pull the robot back. As such, it was hypothesized that the assumptions would hold and interaction force would be bound to 25 N.

It was also expected that subjects would be able to infer the goal to which the ballbot was leading them. In previous work, the robot would direct its turret to point at an intended goal. However, this feature was disabled for the trial conditions.

Lastly, the algorithmic framework for leading was expected to be successful in traversing between goals and maintaining stability. Participants in the trial were also expected to be comfortable with the robot and perceive it as successful in the task. These hypotheses were based on previous work such as that in Section 6.1.

Procedure

Subjects were led to 10 different goals (sampled from the six possible) in the lab set up shown in Fig. 6.14. Throughout the entirety of the trial, an experimenter was within two steps of the

robot in case of any type of failure. After each goal, the subject was handed an Android tablet and asked to choose the goal, of the six, to which they thought they had been led. On a scale of 1-5, they were also asked to answer how comfortable they were with the experience and how successful they thought the robot was in leading them.

The first four goals were unknown to the participant. The remaining six goals were chosen by the participant on the tablet after answering the other three questions. Four of the remaining six goals were executed faithfully to the choice of the goal by the subject, but two goals were "lies." For two goals, a different goal trajectory was executed by the robot, and this was not told to the subject *a priori*. The goals were arranged in pairs as (A-B), (C-D), and (E-F). If the subject chose goal "A" on a "lie" trial, then goal "B" was executed instead. Because subjects were given a free choice of any of the six goals (except the location that they were currently at), some modifications were necessary during experiments. Namely, if a subject was at goal "C" and chose to go to goal "D," then an extra normal goal run was inserted as a "lie" trial would be impossible for that choice. The "lie" trial was simply executing the next time that the user chose a possible goal.

Video was recorded throughout the experiments. Using an Optoforce® 3-axis force gauge connnected to the ballbot's arm end effector, interaction at the wrist was also recorded throughout the experiment. Laser scan, odometry, planned trajectories, localization estimate, and robot diagnostic data were also recorded for each experiment.

After the experiments, each subject completed a post-study questionnaire. The first section assessed their mood and how it had changed from before the trial. The second section asked them to rate statements from 1 to 7 with 1 meaning "Very Strongly Disagree", 4 meaning "Don't Know", and 7 meaning "Very Strongly Agree." After the questionnaire, the participants were asked verbal questions with free response in a recorded interview with the experimenter.

6.3.4 User Subject Trial

Institutional Review Board approval was requested and received to run the study described with able-bodied participants. These subjects were recruited through the Center for Behavioral and Developmental Research online recruitment system. Each subject participated in the trial for approximately 30 minutes.

11 subjects participated in the trial. The data from two subjects were discarded because of a mechanical problem causing inconsistent performance in those experiments. Although data were still collected, the experiments were qualitatively dissimilar from the other subjects runs. Thus, these data were excluded in quantification and analysis of the results.

Of the nine subjects with data used for analysis, there were five female and four male participants. They ranged in age from 23 to 59 years old with a mean age of 32. They averaged a high usage of computers at a self reported 6.9/7 (Never-1, Daily-7), and a low familiarity with robots with a reported 2.5/7 (Not at all-1, Very familiar-7).

6.3.5 Evaluation

The subject trial was a success with the ballbot successfully leading participants and no major problems. Quantitative response from the participants was overwhelmingly positive. The average

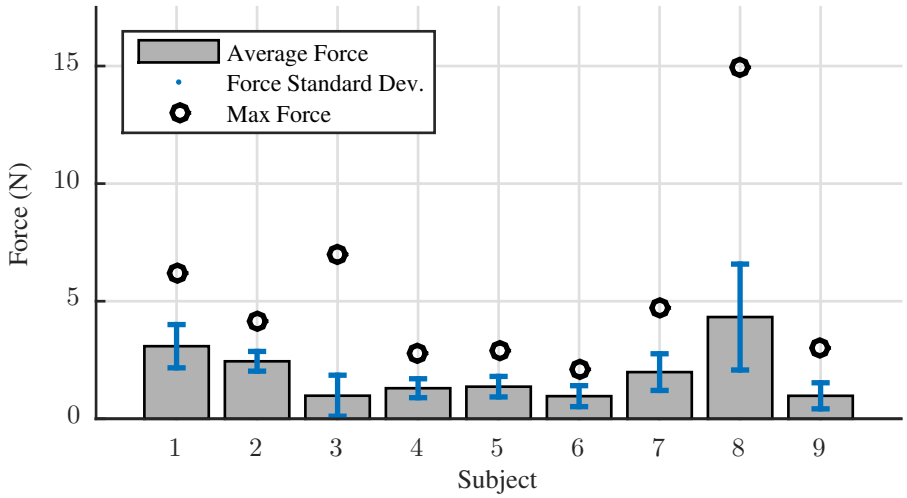


Figure 6.15: Interaction forces during user subject trial on leading. This graph shows the average and maximum forces recorded for each of the nine subjects.

reported comfort of participants via response on the android tablet was 4.73/5, and the average reported success of ballbot was 4.58/5. Note that these values also include the "lie" trials in which the robot led the participants to a goal they were not expecting.

One of the most important results of this subject trial is the force data collected from the ballbot's force gauge. The average force, maximum, and standard deviation for each participant can be seen in Fig. 6.15. The highlight of these data is that the average of all maximum forces is only 6.25 N. This is well below the expectation of a 25 N maximum. The maximum force across all participants is 14.94 N. Of note, it appears that not only does the approach for leading maintain a bounded interaction force, but also the force is much lower than expected.

Next, the subjects also showed a significant capability to determine which of the six goals they were being led to. Among trials in which the ballbot led the participant to a chosen goal (no lie) or an unknown goal (first four trials), the subjects were able to correctly assess which goal the robot led them to 97% of the time (59/61). Of note, some responses were lost due to dropped connections between the android tablet and the logging computer. This problem was apparently random and did not introduce a systematic error.

Interestingly, participants rated the success of leading as significantly lower when the robot was lying to them. After trials where the ballbot led the subject to a different goal than the one they had selected, the average reported success of the robot was 3.5/5. The average success among trials with no lie was 4.8/5. Using the "lie" average as the null hypothesis for a one-sided T test, these results represent sampling of different populations with a p-value of .005.

Overall, participants rated the interaction with the ballbot as very positive and successful. The full results of the post-study questionnaire are shown in Table 6.2. Significant responses are denoted with an asterisk ($p < 0.05$). Multiple asterisks denote responses with more significance and denoted in the legend.

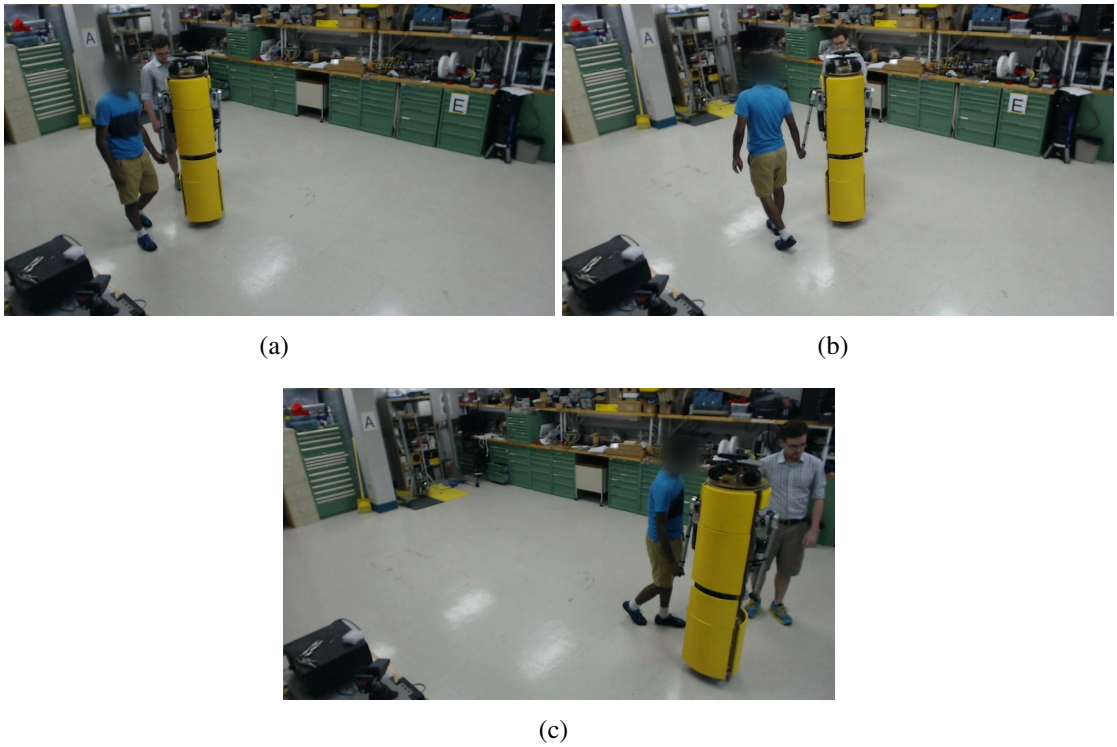
Question	Average Response
I would feel comfortable standing next to Ballbot while it stands still.	6.78*
I would feel comfortable standing next to Ballbot while it moves slowly.	6.67***
I would feel comfortable standing next to Ballbot while it moves quickly.	5.56**
I would feel comfortable working around Ballbot.	6.22***
I would feel comfortable sitting next to Ballbot.	6.22***
Ballbot moves in a graceful way.	5.56**
Ballbot moves in an intelligent way.	5.89**
It would hurt if Ballbot ran into me.	4.11
I could push Ballbot out of my way.	4.33
I could tell which direction Ballbot was about to move.	5.89***
I would be comfortable walking alongside Ballbot.	6.44***
I would be comfortable passing Ballbot in a hallway.	6.56***
I would feel comfortable leading Ballbot by the hand.	6.44***
I would feel comfortable letting Ballbot lead me by the hand.	6.33***
I could tell where Ballbot was leading me.	6.11***
Ballbot successfully led me to locations	5.56***
Ballbot used too little force to lead me.	3.44
Ballbot used appropriate force to lead me.	6.11**
Ballbot used too much force to lead me.	2.00*
Ballbot could carry household objects.	5.11
Ballbot could help me carry objects.	5.22*
I would feel comfortable carrying a card table with Ballbot.	4.67
I would like to have a Ballbot in my workplace.	6.44**
I would like Ballbot to lead me around a building	6.00***
I would be comfortable with Ballbot in my home.	6.22***
I would like a Ballbot in my home.	6.22***
I think Ballbot is safe.	5.33**
Ballbot is too big. If it were smaller, I would be more comfortable with it.	4.78
Ballbot is too fast. If it were slower, I would be more comfortable with it.	1.89*
I would feel more comfortable around Ballbot than around an industrial robot arm.	5.00
I would feel more comfortable around Ballbot than around a humanoid walking robot.	5.11*
I would feel more comfortable around Ballbot than around a 4-wheeled robot.	5.00*

Table 6.2: Subject Questionnaire Responses, $H_0 = 4$ (*= $p < .05$, **= $p < .01$, ***= $p < .001$). Note that all p values are calculated from one-sided T tests with a null hypothesis of 4 (Dont Know). Average responses below 4 are calculated as one-sided below, and one-sided above for averages above 4.

Qualitative Evaluation

Along with the qualitative results already discussed, the results elucidated interesting qualitative and anecdotal findings from the subject trial as well. During the trial, the ballbot's yaw controller

was not active. This allows the robot to yaw freely and gave the users freedom to choose their orientation relative to the robot. Overall, participants did not show a strong preference toward respect of the robot's "front." In fact, many subjects changed their relative position to the ballbot between subsequent runs. An example of this can be seen in Fig. 6.16.



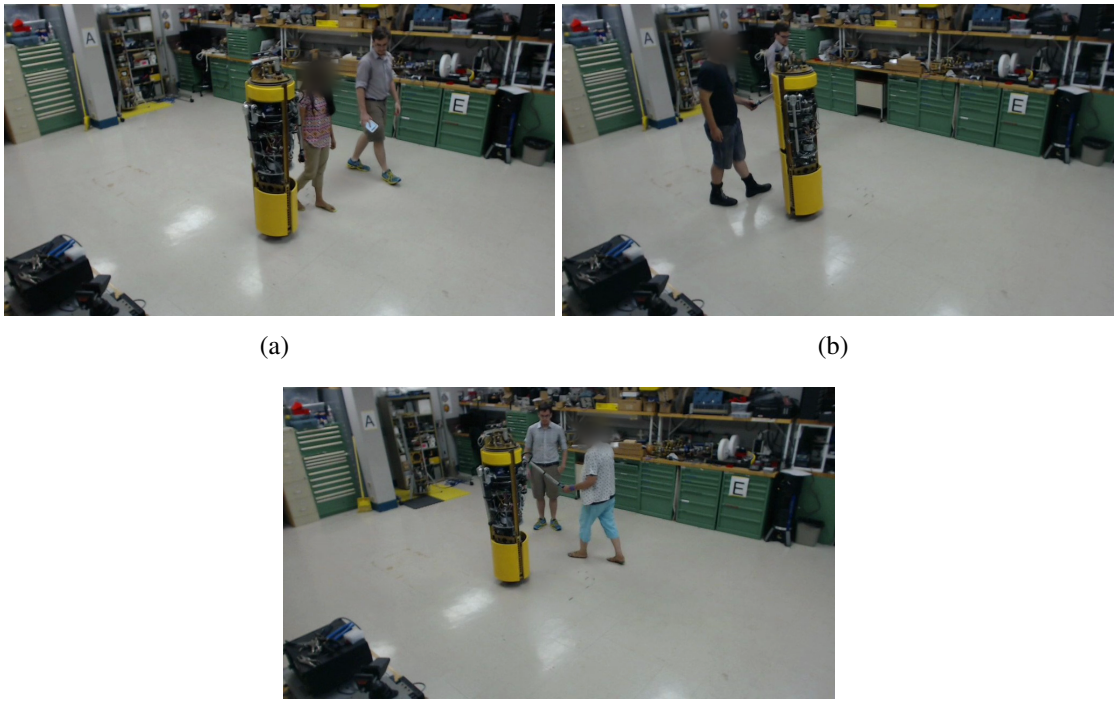
(a) (b)

(c)

Figure 6.16: The ballbot leading a participant in user subject trial. Three different goal executions are shown, with the subject choosing a slightly different spatial relation to the robot each time. Faces have been blurred per compliance with the IRB for this study.

We also analyzed the position of the subjects' arm relative to the robot. Interestingly, there was no single configuration common to all participants. Examples of different relative arm positions can be seen in Fig. 6.17.

After the experiments and questionnaire, subject were interviewed and recorded. Overwhelmingly, participants responded positively to, "Do you think the robot seems helpful," and negatively to, "Do you think the robot seems frightening." One participant answered the question, "Was it easy to walk with the ballbot," with the response, "Oh yeah, it was fine. [There was] a little adjustment phase on the first run, but I got used to it really quick." When asked what kind of tasks a participant could envision the ballbot performing in the future, they responded, "Help elderly people or people with Parkinson's walk. It could also help them carry items."



(a) (b)

(c)

Figure 6.17: Three different participants in the leading user subject trial. Each subject chooses a different arm position relative to the ballbot's arm. Again, faces have been blurred per compliance with the IRB for this study.

6.3.6 Future Applications and Experiments in Assistive Leading

The subject trial presented in this section was successfully executed and completed. The hypotheses were all confirmed and some expectations even exceeded. These novel experiments represent proof-of-concept for a physical assistive navigation technology using the ballbot.

The most pronounced limitation in interpreting the user study data is that all participants were able-bodied individuals. Elderly, disabled, and visually impaired people stand to gain the most benefit from this techonology. As such, one future direction is completing the same study with one or all of these populations. IRB approval is significantly more difficult to attain for such a study, but the success of the trials present should help make a strong case.

Figure 6.18: The ballbot leading a person through a doorway by the hand.

The trial data presented also involved leading subjects through an open room in mostly straight-line paths. Another avenue of future exploration is interacting with more complex environments and obstacles. Fig. 6.18 shows the ballbot leading a researcher through a doorway. Because the core of the leading system utilizes the navigation algorithms presented in Chapter 5, these capabilities already exist. Before testing this functionality with human subjects, however, some thought would need to be given to relative positioning of the ballbot and subject. For example, the robot would need to make sure the subject is behind it while going though a tight doorway even if the subject prefers to stay by the robots side, as in Fig. 6.17(a).

Lastly, another future area of research is incorporating conversation into leading. Fig. 6.19 shows a complete state machine for both teaching ballbot the names of new locations and requesting the robot to lead someone by the hand to one of those locations. This software system relies on Google Speech® for speech-to-text and the ROS package SMACH for the state machine container. The robot also gives the participant information through Cepstral® text-to-speech. Some initial experiments have been conducted, and the system shows potential. A human subject trial would be required to identify and optimize the best states and responses for the system.

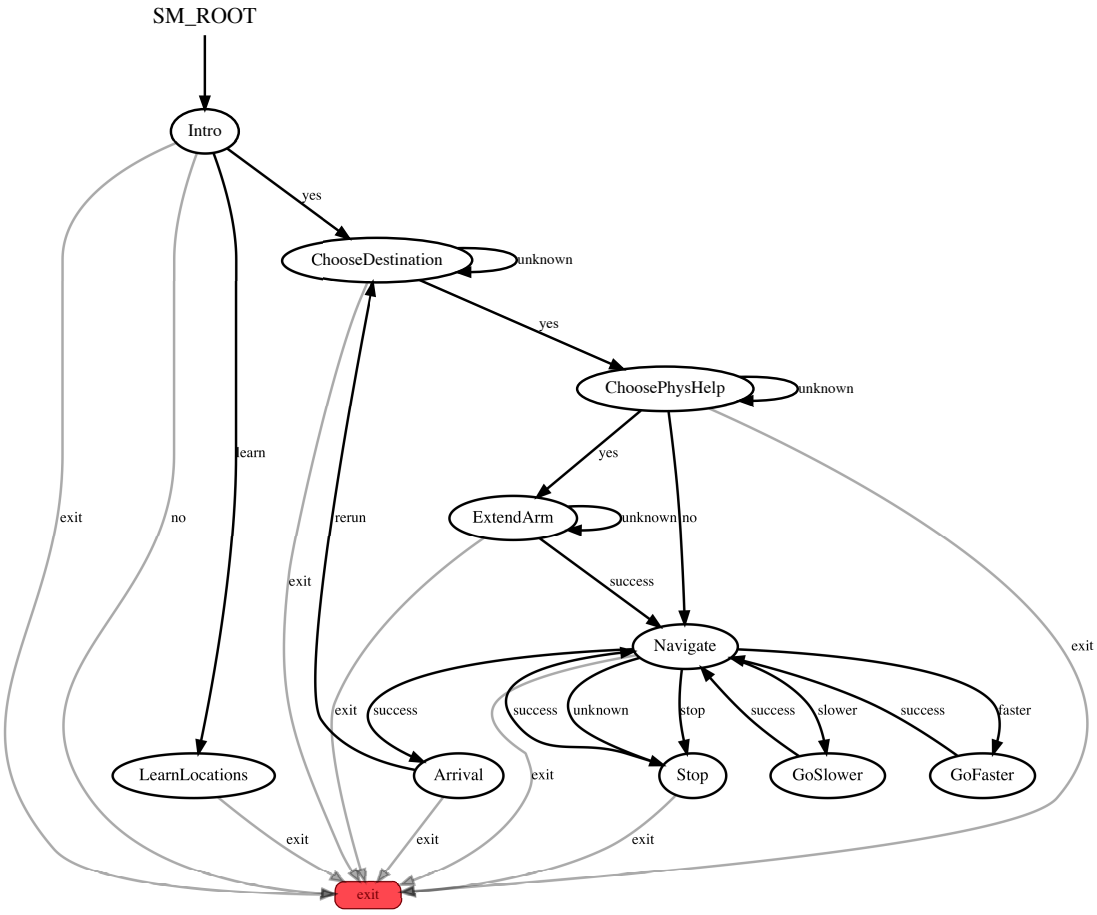


Figure 6.19: State machine diagram for conversation based navigation using the ballbot. This architecture leverages the ROS package SMACH.

6.4 Toward Contact-Informed Crowd Navigation

Another interesting extension of both ballbot navigation and physical interaction is the problem of navigation through dense crowds. This section explores a possible method for crowd navigation through tactile interaction. Namely this method attempts to use dynamics as means to detect collisions and then use that information to inform better planning. The ballbot is an ideal platform for such an endeavor, as it can safely collide with objects and people with bounded force.

As discussed in Chapter 2, there has been a substantial focus of navigation in crowds in recent years. All the methods investigated rely on models of people and the interaction the robot will have them. Overall, most of these methods rely on "Cooperative Collision Avoidance." In other words, these methods plan trajectories that will not collide with anything, given the model of how people will react to the movement. Many of these strategies have shown potential and indeed some success in navigating crowds. However, all such methods rely on the assumption that either cooperative collision avoidance will occur, or a collision free path will present itself at some time. In a very dense crowd with relatively stagnant movement, these assumptions may break down. An example of a dense crowd is shown in Fig. 6.30.

To achieve successful navigation in dense crowds, contact and indeed collision could be necessary. As such, this section explores a means to collide safely and use the information for planning. To our knowledge, this is the first attempt at such a system.

One exception is the preliminary work by Shreshta et. al described in the Chapter 2 [Shrestha et al., 2015]. However, their approach requires a collision free path after the human moves following inducement. Situations such as those seen in Fig 6.30 make this approach impossible. Similar situations arise in cocktail parties, hospitals, or even hallways of businesses in high-traffic times.

6.4.1 Planning

As in Section 6.3, we use the same general planning hierarchy as demonstrated previously, discussed in Chapter 5. At its core, the paths are planned on two-dimensional, occupancy grid-based costmaps. Prior work then generates dynamic, feasible trajectories free of collision. These trajectories are planned to be collision-free because the cost associated with travelling through any observed obstacle is infinite. In this section, we would like to challenge that notion by using a finite cost to represent non-static obstacles in an environment. Using the force detection and estimation strategy from the previous section, the cost of traversal will then be updated based on experienced impediment.

When a collision occurs, the ballbot will likely not be able to track the desired global trajectory with acceptable accuracy. As such, the same state assessment and replanning strategy described in Section 6.3.1 was used. As with leading, the robot can be in collision indefinitely without its lean angle growing unbounded. The difference between the use of this method for crowd navigation, however, is that the costmap will be changing during that time in contact. This can cause the ballbot to replan a new trajectory around an object (or person) that is not moving.

Layered Costmap

This formulation of the crowd navigation problem relies on a multi-layer costmap [Lu et al., 2014]. The costmap starts with a layer of known static obstacle based on previous mapping of the experimental space. The next layer, associated with dynamic obstacles, uses a prior to set a finite cost on observations from sensors. From this updated costmap, a path to goal can be planned and initiated. The proposed method of crowd navigation uses another layer that starts with no cost. This "impediment" layer then has cost added to it incrementally when and where the robot detects that it is in collision.

During execution, collisions are detected via Eq. 6.8 and forces estimated. By setting a maximum lean angle for the ballbot and traversing at a slow speed through high-cost areas, the max collision force should be bounded by Eq.6.1 assuming quasi-static interaction. A quasi-static assumption was shown to be sufficient for force estimation, even in the case of fast dynamic assistance in Section 6.2.

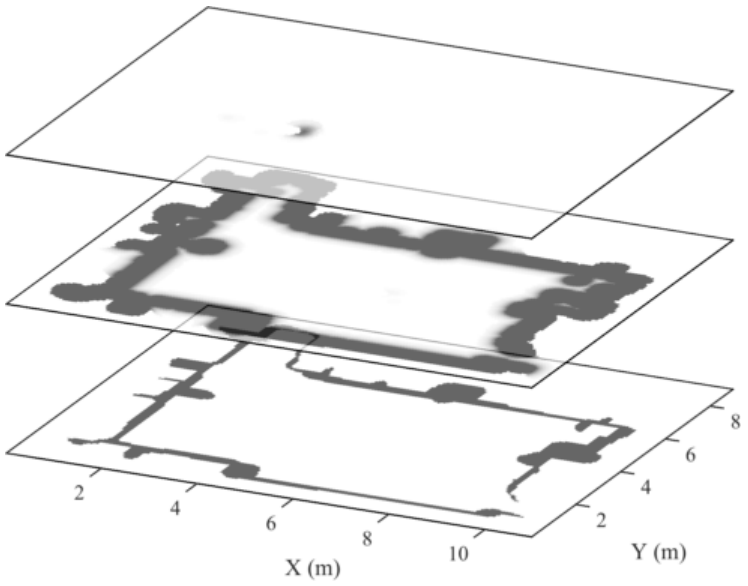


Figure 6.20: Layered Costmap with three layers. The lowest layer is the static map. The second layer is an obstacle inflation of the static map. The third layer is the "impediment" layer, with cost added if and when the ballbot collides with objects.

The static map layer uses an obstacle inflation. The navigation presented in Chapter 5 is the same concept without the additional "impediment" layer. Fig. 5.11(b) shows the costmap with infinite cost obstacles in blue.

Since a collision-free path should have a lower cost even without infinite cost obstacles, this modified system should still prefer a collision free path should one exist. If none exists, then the ballbot will navigate with possible contacts, updating the costmap along the way. The ballbot's inherent capability to bound contact force enabled this forceful exploration.

Before planning for collisions, a preliminary step is detecting collisions. Ideally, this can be achieved without the use of force or contact sensors. Estimating external force is certainly attainable in the static case, as a constant force in static equilibrium should yield a constant lean

angle as can be seen from Eq. 6.1. In this section, a method for detecting and estimating force in the dynamic case is presented.

6.4.2 Sensing External Forces on Ballbot

As in prior work, we consider a planar, decoupled ballbot model for ease of modeling. Fig. 6.21 shows the planar model with an external force from collision acting on the ballbot.

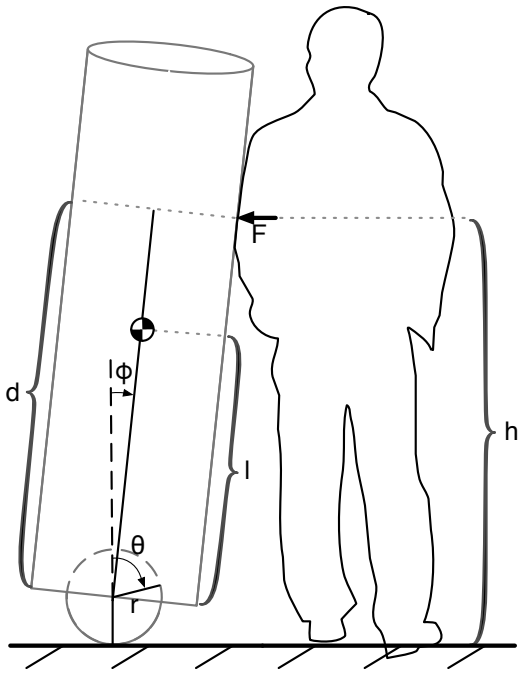


Figure 6.21: The ballbot in contact with a person, experiencing a force. As in Fig. 3.3, this is a planar model with state variables lean angle ϕ, ball angle θ. Here, F is the interaction force, acting at height h. M_b is the mass of the ballbot body. M_w is the mass of the ball (wheel).

The same nomenclature is used for this model of the system as in the previous planar analysis. However, the external force, F was added to the model. The force is assumed to act in the horizontal direction. In the case of a zero lean angle ϕ, any force F will act on the center of mass of the ballbot, causing it to move. A non-zero desired lean angle can result in static equilibrium with a nonzero force, as discussed previously.

The ballbot does not have force or contact sensors on its outer shell, and it is desired to detect external forces by observing the time evolution of the system. Toward this end, let σ be a scalar defined as

$$\sigma(t) = k_D[E(t) - \int_0^t (\dot{q}^T \tau + \sigma) ds - E(0)]. \tag{6.8}$$

σ is the collision detection signal [Luca et al., 2006]. This concept derives from collision detection in robotic arms without torque, force, or contact sensors. Here, k_D is a parameter

chosen to stabilize detection; q is vector of state variables, E is the total energy of the system, and τ is the control input. Although this formulation was proposed for fully actuated systems, it is sufficiently general to apply to the ballbot. Note that

$$\dot{\sigma(t)} = -k_D\sigma + k_D\dot{q}^T\tau_K. \tag{6.9}$$

Eq. 6.9 shows that σ is a first order stable linear filter driven by the collision force. τ_k is the torque associated with the collision force by

$$\tau_K = J_K^T(q)F_K, \tag{6.10}$$

where F_K is the external collision force.

This method has been shown to work well with robot arms, especially with a good estimate of the actual torque exerted by motors on the links. The ballbot's IMB drive mechanism has significant friction, a factor which distorts estimates of the torque magnitude being transmitted to the ball. Therefore, instead of using the collision detection signal as presented in [Luca et al., 2006], our experimental method takes advantage of the ballbot's internal system dynamics, Eq. 4.5.

Similar to the collision signal method, an energy term is added into the equations of motion from the Lagrangian. In the case of the ballbot's internal system dynamics, we can simply change the left hand side from zero to an energy term. This represents energy being injected or stolen from a collision:

$$\sigma = (\alpha+\beta)\ddot{\theta} + (\alpha+\gamma+2\beta)\ddot{\phi} - \beta\phi\dot{\phi}^2 + \frac{\beta g\phi}{r}. \tag{6.11}$$

The internal system dynamics contains some terms that are not directly measured and are instead found by finite differencing. Therefore, significant filtering is required. The internal system dynamics are split into four terms for the purpose of filtering with different time constants as follows:

$$\sigma = \underbrace{(\alpha+\beta)\ddot{\theta}}_{z_1} + \underbrace{(\alpha+\gamma+2\beta)\ddot{\phi}}_{z_2} - \underbrace{\beta\phi\dot{\phi}^2}_{z_3} + \underbrace{\frac{\beta g\phi}{r}}_{z_4} \tag{6.12}$$

The terms are filtered with a first order digital low pass filter:

$$z = (\alpha z_t) + ((1-\alpha)z_{t-1}) \tag{6.13}$$

with

$$\alpha = \Delta t/(\tau + \Delta t). \tag{6.14}$$

The time constants, τ, for the various terms were found empirically and are list in Table 6.3:

Experimental Platform and Validation of Force Estimation

The proposed method of crowd navigation necessarily involved collision with people and objects. Because this involves more risk with the ballbot than any previous line of inquiry, a new robot

Term	Components	Time Constant (ms)
z_1	$(\alpha+\beta)\ddot{\theta}$	200
z_2	$(\alpha+\gamma+2\beta)\ddot{\phi}$	400
z_3	$\beta\phi\dot{\phi}^2$	200
z_4	$\beta g\phi/r$	0

Table 6.3: Force Estimate Low Pass Filter Time Constants

Figure 6.22: Testing force estimation with a force gauge.

platform was used for experimentation. The robot is a much smaller ballbot at 1.2 m tall and 16 kg. The robot is tentatively called shmoobot and can be seen in Fig. 6.22. In the figure, the robot is station-keeping, using all the same controllers and software as previously described for the ballbot. It has slightly different gains to accommodate its smaller size and mass. While station-keeping, the robot was pushed by an experimenter from many different directions. The quantitative results of this experiment are shown in Fig. 6.23.

Of note, the force estimate has a strong response to each push from the force gauge. Although the magnitudes are not equivalent, the peaks are roughly proportional. A high fidelity reproduction of the forces is not necessary, as the main goal of this estimator is simply to determine whether collision occurred and gain the rough magnitude. Also, the estimator has a roughly .3 s phase lag. This is due to the low pass filters described in Table 6.3. The low-pass filters are necessary as quantities such as $\ddot{\theta}$ are twice finite differenced and exhibit significant noise.

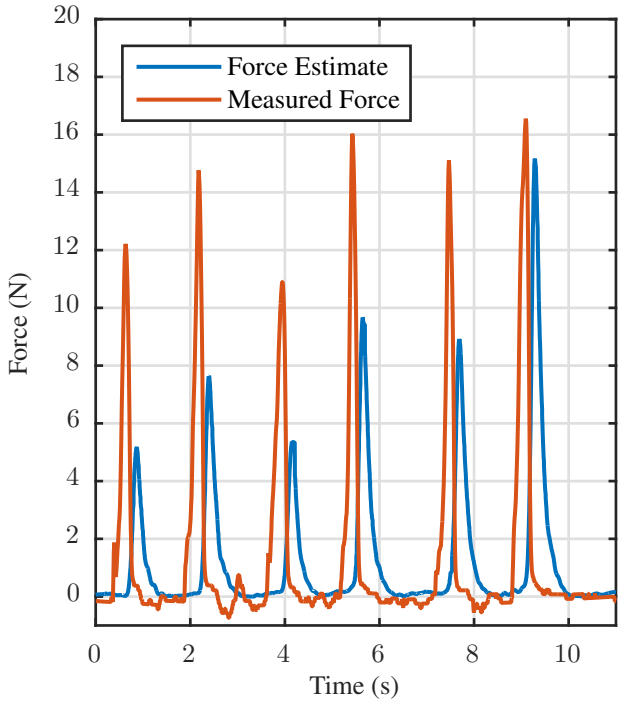


Figure 6.23: External force estimate σ compared with ground truth from force gauge. This experiment was performed with the small ballbot.

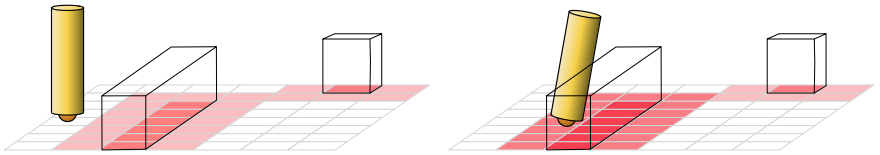
Impediment Estimation

The proposed method of actively and forcefully investigating the resistance to navigation requires a number of components. External force detection and estimation must be robust and running continuously during navigation. The expected evolution of the "impediment" costmap layer is shown via a time evolution of the proposed method in Fig. 6.24

This demonstrates how the costs associated with obstacles might evolve after contact. (a) shows the ballbot and the costmap with costs mapped to the intensity of the color red. Both observed obstacles have equal cost and have an inflation radius of slightly lower cost. The granularity of this example is simply to illustrate how the algorithm functions. (b) shows the ballbot in contact with the first obstacle. Because this obstacle is immovable, the cost associated with this obstacle and its inflation have increased. In practice, this will initiate a replanning event. After replanning a new trajectory, (c) shows the ballbot in contact with a movable obstacle. The force applied by the robot causes the object to move in (d), and therefore, the cost does not increase.

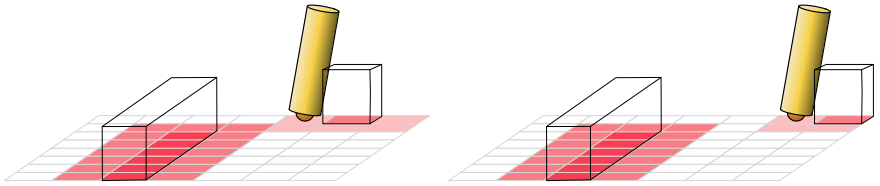
From Force Estimate to Costmap Update

With the ability to estimate external forces acting on the ballbot, the next step was modifying the impediment costmap layer. As this was the first attempt at such a system, simplicity is technically



(a) The ballbot, in contact with no obstacles assigns a cost to the two obstacles and adds inflation

(b) In contact with the first obstacle, but unable to move it, the ballbot increases the cost and inflation.



(c) Now the ballbot is contact with a movable object.

(d) Because the object moved, the cost is not increased. The associated cost and inflation moves with the obstacle.

Figure 6.24: Example of "Impediment Costmap Layer" evolution with one immovable obstacle and one movable obstacle. Costs are shown in red.

desirable. To this end, the costmap is modified by incrementally adding cost as a gaussian "blob" offset from the ballbot in the direction of collision. The direction can be estimated because the force estimate, σ, is calculated in both decouple planes (X and Y) for the ballbot.

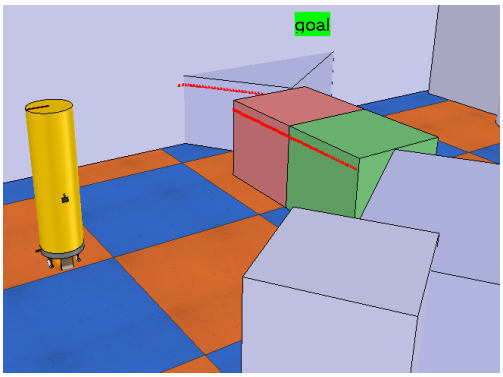
The parameters affecting the behavior of this costmap modification were empirically tuned on the system. These tuned parameters are shown in Table 6.4. The σ estimate threshold is the minimum value, which must be achieved to add any cost to the layer. The Cost Scale Factor is simply the multiplier for the center value of the gaussian, multiplied by σ. The costmap values are integers from 0 to 255, with 255 denoting a "lethal" obstacle. Because σ is continuously estimated at the control frequency, it must be throttled for costmap calculation and numerical stability. As such, updates to the costmap are throttled to the Maximum Rate. The Blob Offset Distance is the distance from the center of ballbot at which the gaussian center is placed.

Parameter	Value
σ Estimate Threshold	0.23
Cost Scale Factor	16.0
Maximum Rate	5 Hz
Blob Offset Distance	0.4 m

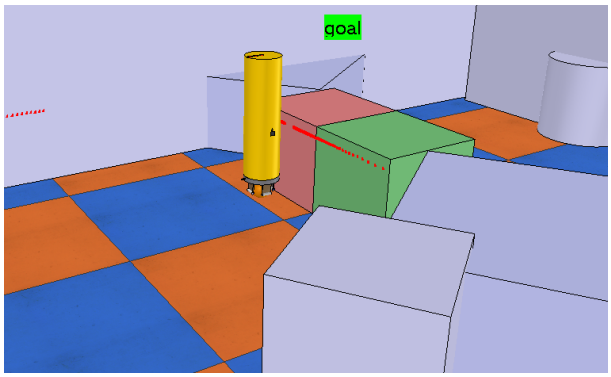
Table 6.4: Force Estimate to Costmap Update Parameters. These parameters were used for experiments with the small ballbot.

6.4.3 Simulation

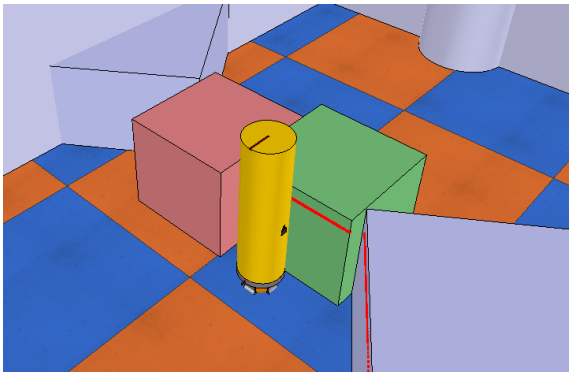
In parallel with testing the force estimation on the real hardware, as simulation environment was developed using the V-Rep [Rohmer et al., 2013] simulator. This system integrates with ROS, and uses all the same message types and infrastructure as the real system. An experiment testing the feasibility of external forces inducing cost can be seen in Fig. 6.25.



(a) Shmoobot on the left side of the scene is given a goal on the right side, past the red (immovable) box and the green (movable) box.



(b) Shmoobot executes its planned trajectory, colliding with the immovable box.



(c) After modifying its costmap based on the forceful interaction with the red box, shmoobot replans and begins to push the green box out of the way.

Figure 6.25: V-Rep simulation environment, testing the force-to-cost software stack.

Although this simulation cannot reproduce forceful interaction with high fidelity, it is still very important. The first goal of this simulation was to verify the software stack. Next, reasonable reactions needed to be verified and validated when collision-free paths exist. The full software stack with updates to the costmap from collisions could be tested with it, but real experiments are necessary. Currently, the simulated ballbot's dynamic performance is not similar enough to the real robot for force estimation. To circumvent this, a "force sensor" shell was placed in simulation to measure the exact force in x and y, thereby enabling testing of the rest of the software stack and debugging before testing on the shmoobot hardware. In the future,

V-Rep could prove very useful for future experiments as it can simulate humans, both standing and walking in proximity to the robot.

6.4.4 Experiments

To test the efficacy of this planning strategy, some preliminary experiments were run with the shmoobot. The purpose of these tests was to evaluate the feasibility of the complete software stack. Tests were conducted with static obstacles (unknown to the robot) as well as with the experimenter as a static obstacle.

Fully realizing the capability of navigating through crowds with this method requires many more tests with real human subjects. Such tests require IRB approval and more infrastructure development. The work presented here demonstrates preliminary work towards eventually accomplishing this goal.

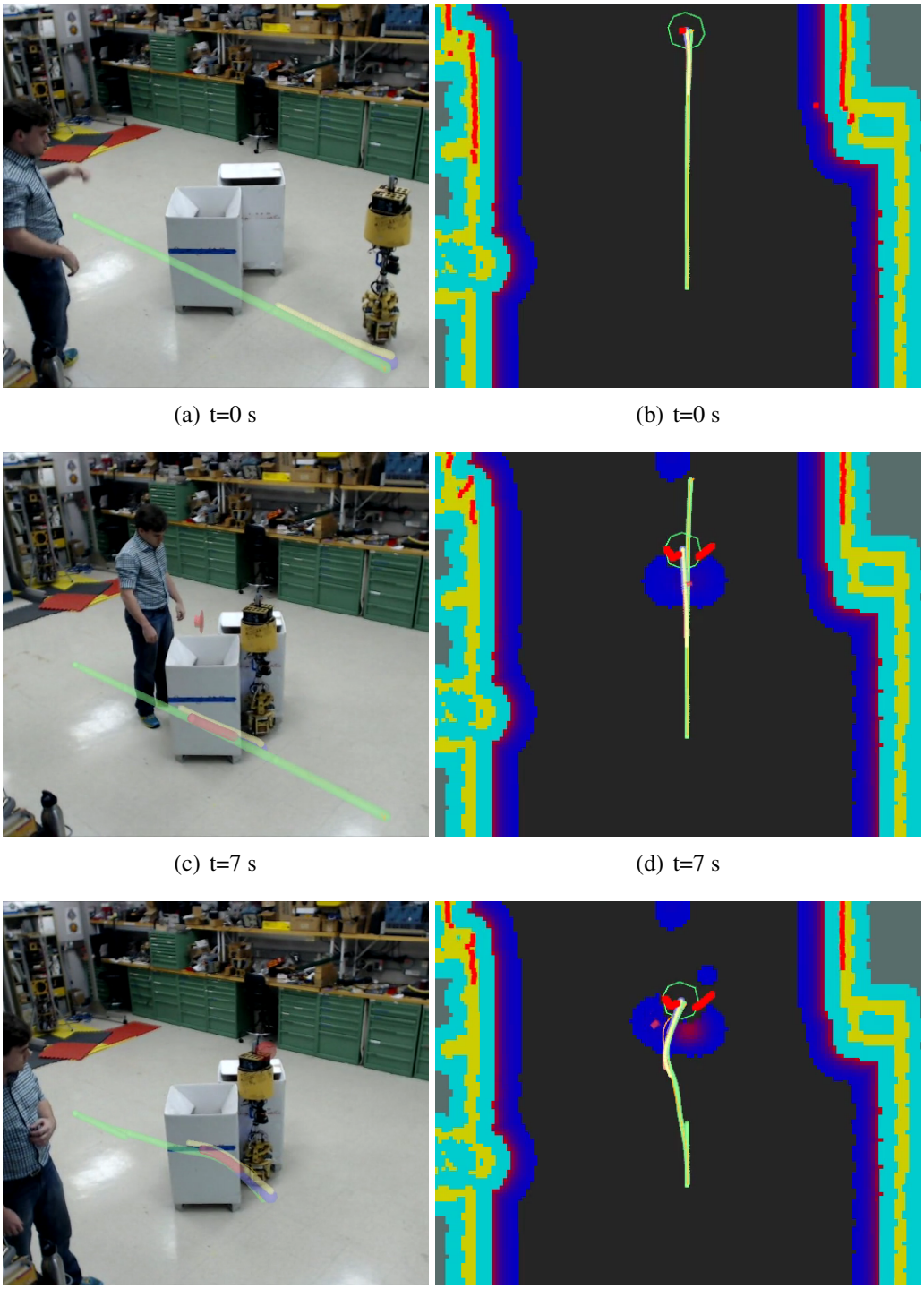
Two experimental results are shown in Figures 6.26, 6.27, 6.28, and 6.29 over the next four pages. For all of these experiments, although a laser range finder was used for localization, obstacles were not added to a costmap layer based on laser returns. Instead, all obstacle information was inferred through contact. In a true crowd navigation situation, this would likely not be the case. However, the purpose of these experiments was to test the efficacy of the Impediment Layer as a means to achieve navigation through tactile interaction.

Figures 6.26 and 6.27 show an experiment using two boxes as static obstacles. The boxes fully block the straight-line path of shmoobot, trapping it in a local minimum. Each snapshot view of the scene on the left has a corresponding top-down view of the current costmap representation on the right. In this costmap, blue represents a very low cost. Red represents moderate cost, and cyan represents the maximum cost for the impediment layer.

Figs. 6.26(a) and 6.26(b) show the beginning of the experiment with the robot unaware of any obstacle. It was given a goal in the upper left corner of the scene, and it planned a straight-line path to goal. Figs. 6.26(c) and 6.26(d) show the robot making first collision with the obstacles and adding cost to its costmap. The bright red dots are the laser scan returns from the obstacle. These are shown to help localize the obstacle in the costmap for visualization. Notably, the estimated force is shown in both views as a three dimensional red arrow. At 13 seconds into the experiment, Figs. 6.26(e) and 6.26(f) show that the costmap has acquired sufficient cost to trigger a replan. Fig. 6.26(f) shows a new trajectory to the left of the estimated cost. This new path does not deviate enough from a straight line yet to escape the local minimum.

Figs. 6.27(a) and 6.27(b) show another replanning event, this time triggering a trajectory attempt on the other side of the boxes. In Figs. 6.27(c) and 6.27(d), the accumulated cost has now grown enough that the replanned trajectory can clear the obstacles. Figs. 6.27(e) and 6.27(f) show the robot executing a now collision free trajectory after learning about the obstacles through forceful interatcion. Some erroneous blue dots exist in the costmap. Again, blue represents a very low cost, and the system demonstrates the presence of some noise.

Figures 6.28 and6.29 show a similar experiment using a box and the experimenter as static obstacles. As with the previous experiment, this traps the robot in a local minimum, forcing it to learn the costmap through forceful interaction.

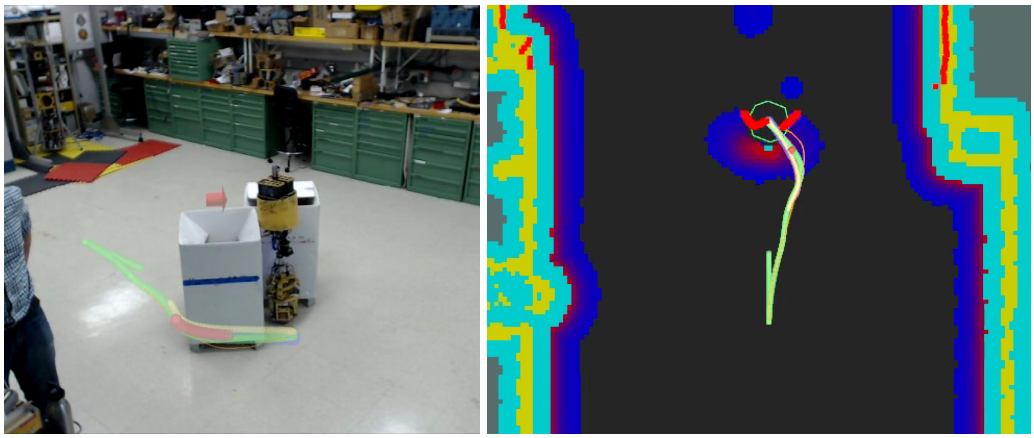


(a) t=0 s (b) t=0 s

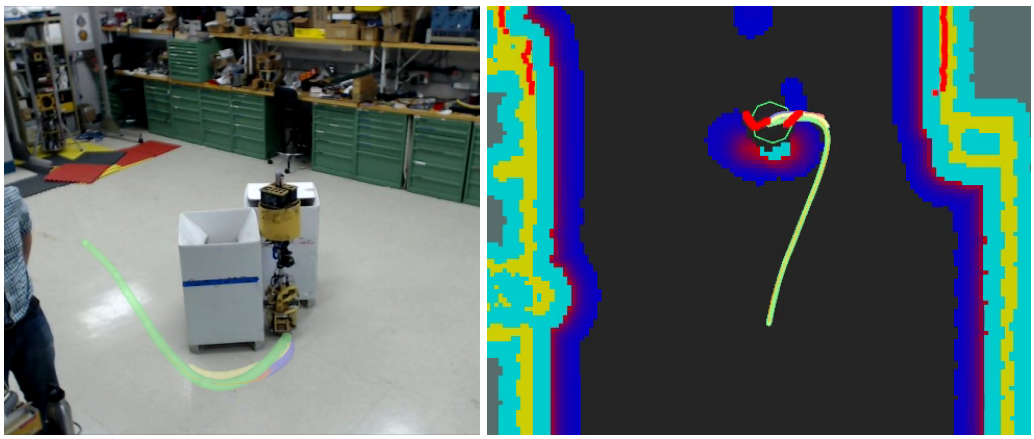
(c) t=7 s (d) t=7 s

(e) t=13 s (f) t=13 s

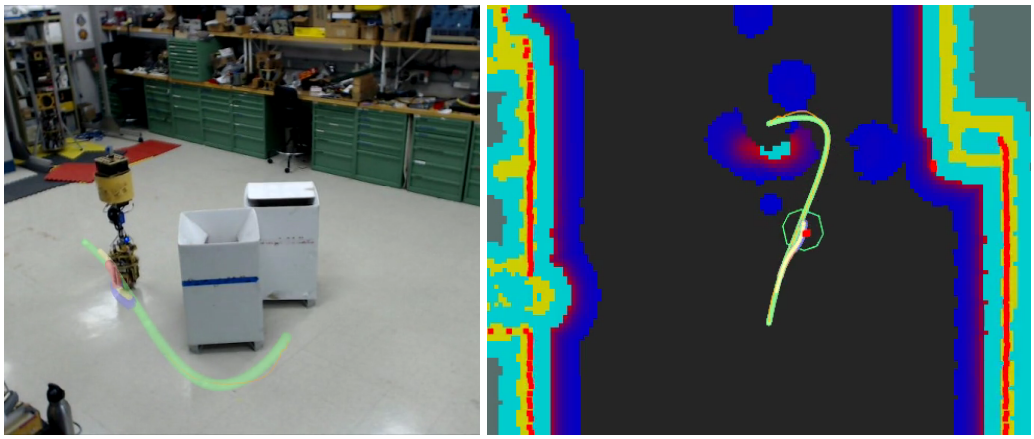
Figure 6.26: Experiment to test collision and costmap alteration on a local minimum. Camera view of the experiment is shown on the left. Top-Down Costmap view is shown on the right. The time of the snapshot relative to the experiment is shown in the caption.



(a) t=20 s (b) t=20 s

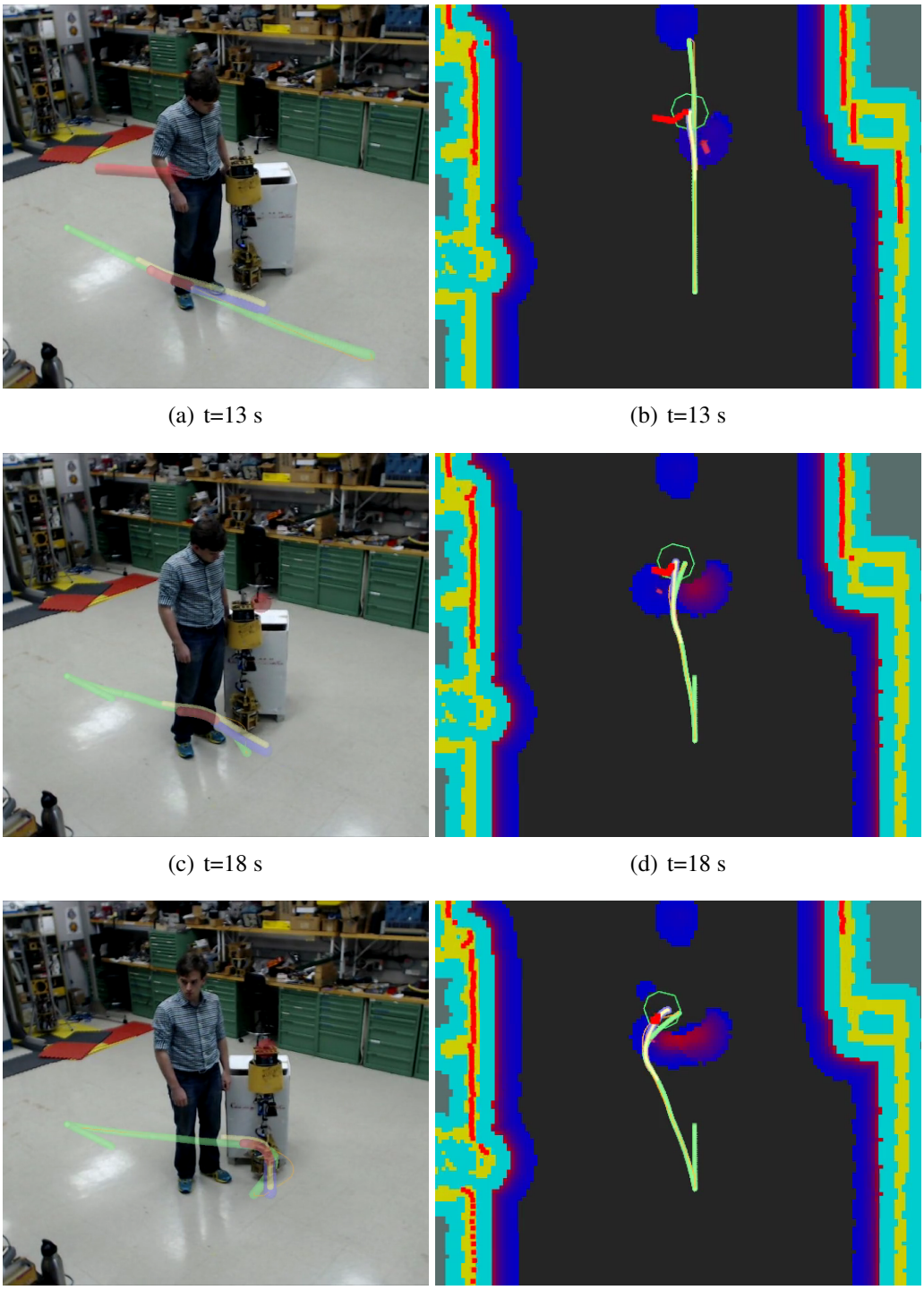


(c) t=25 s (d) t=25 s



(e) t=36 s (f) t=36 s

Figure 6.27: Camera and costmap views of local minimum experiment. Continued from Fig. 6.26

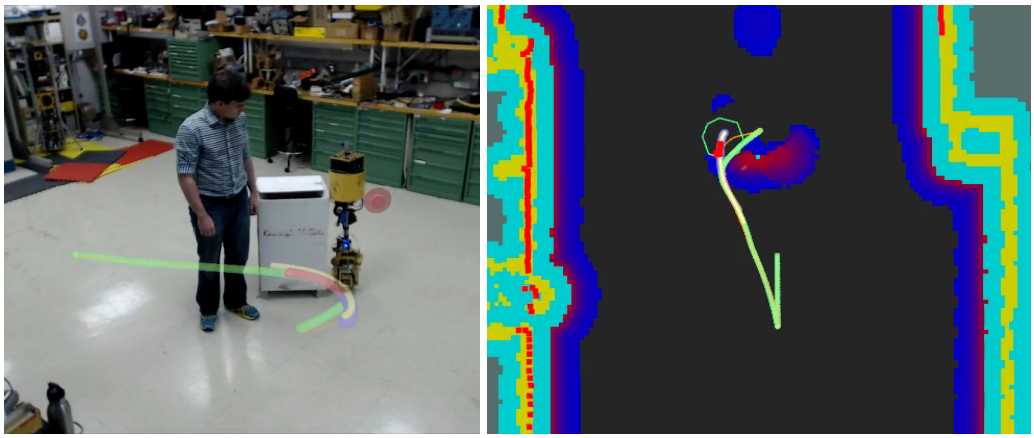


(a) t=13 s (b) t=13 s

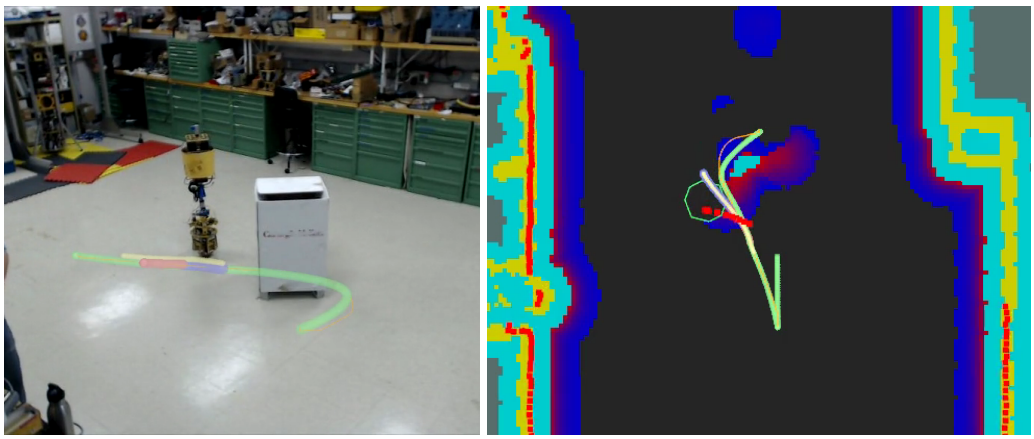
(c) t=18 s (d) t=18 s

(e) t=24 s (f) t=24 s

Figure 6.28: Experiment to test collision and costmap alteration on a local minimum with a person. Camera view of the experiment is shown on the left. Top-Down Costmap view is shown on the right. The time of the snapshot relative to the experiment is shown in the caption.



(a) t=39 s (b) t=39 s



(c) t=47 s (d) t=47 s

Figure 6.29: Camera and costmap views of local minimum experiment. Continued from Fig. 6.28

6.4.5 Limitations and Future Work

The strategy for collision estimation and costmap layer modification presented in this section represents a step toward toward tactile navigation through very dense crowds. The experimental results show that this system is capable of detecting collisions through knowledge of dynamics, and using that information to better plan trajectories through space. By estimating these collisions, no external force sensor or tactile skin is required.

Because this strategy relies on a layered costmap, it could be augmented to incorporate other crowd navigation algorithms in addition. The impediment layer could work in tandem with a cooperative collision avoidance layer. Combining these strategies could create a system capable of navigating any crowd density.

Figure 6.30: Carnegie Mellon School of Computer Science Graduate Student "TG." These events are hosted by Dec5, the graduate organization for SCS. With hundreds of attendees in a small space, these present a very challenging crowd navigation scenario.

The next step of testing should be moving amongst people. Fig. 6.30 shows a crowded event, which could potentially be used for experimentation (a TG hosted by Dec5 the graduate student organization for the School of Computer Science). These events present an opportunity to test this system and evaluate whether the inherent assumptions of the strategy fail or succeed. Such experiments would require planning and IRB approval.

Chapter 7

Conclusions and Future Work

The body of work presented in this thesis demonstrates the use of the ballbot as a balancing mobile robot capable of useful and interesting physical interactions and establishes its potential to benefit people. Its form factor and balancing capability make it useful in a variety of scenarios, including physically assisting and leading individuals. This work also involved development of methods for ballbot-mediated navigation in human environments.

The constituent work of this thesis has resulted in a few key insights about navigation and forceful interaction with the ballbot:

System representation can mitigate or eliminate optimization.

Formulating the ballbot as a differentially flat system greatly eases the computational requirements for planning and has enabled the system to generate fast, responsive, and feasible trajectories. Furthermore, Chapter 5 showed that a clever choice of representation for trajectory optimization can make an already efficient QP into an even better linear system problem.

The ballbot is a very capable force-controlled robot through modulation of lean angle

Chapter 6 presented work on helping people out of chairs, leading people by the hand, and sensing the world through collision. All three of these capabilities rely on ballbot as force controller and sensor through control of lean angle and analysis of dynamics.

Next, this chapter lists the contributions of this work as well as some possible future directions of the ballbot project.

7.1 Contributions

The work presented in this thesis has yielded the following contributions:

The formulation of the ballbot as a differentially flat system

Chapter 4 presented a representation of the ballbot as a differentially flat system. This representation greatly reduces the complexity of generated feasible ballbot trajectories by integrating the constraints of the dynamics into the flat output.

Tractable collision-free navigation for the ballbot in cluttered environments

Chapter 5 extended the flatness-based ballbot model to multi-waypoint trajectories. We explored the optimization and integration of these trajectories with a real-time planning system we developed.

Constant-time replanning for localization updates and disturbance rejection

The replanning system described in Section 5.4 allows the ballbot to use localization information in a useful way while still maintaining smooth feedback control from odometry information. To facilitate efforts to lead people by the hand, Chapter 6 describes our work developing novel replanning approaches and collisions with obstacles.

Performance evaluation and real-time implementation of planning method

The navigation capabilities presented in Chapter 5 were implemented and tested on the ballbot. Using this method the robot was able to navigate through obstacles, both static and dynamic. It was also able to execute long trajectories through a building.

Methodology and demonstration of the ballbot applying large assistive forces

With data from the user study presented in Chapter 6, the ballbot was able to demonstrate assistance in helping a person out of a chair. This required application of over 120 N.

A stable method for leading people by the hand with a balancing mobile robot

Section 6.3 presented a method for leading people by the hand. The system was proven robust through the first user subject trial of its kind, using a ballbot with significant physical interaction.

An algorithm for detecting and estimating external forces applied to the ballbot

Section 6.4 showed a method for detecting external forces on the ballbot. This method was tested and compared to ground truth data from a force gauge.

Cost estimation and adjustment through forceful interaction during navigation

Estimated external force was used to modify a costmap and used for the purpose of navigation. This capability was demonstrated in discovering static obstacles through forceful interaction. This work is intended as step toward navigation through dense humans crowds using collisions as an information source.

This thesis has also yielded a number of contributions regarding both hardware and software. These contributions benefit the project as a whole and will potentially be used by future students working on the ballbot:

ROS integration to the ballbot software system

An early contribution to the project was integrating the ballbot system with ROS. This has enabled many new capabilities for the system through easy interfacing with open-source software. Namely, localization and mapping through Hector SLAM, low-level planning infrastructure through the ROS Navigation stack, and visualization via Rviz enhanced trouble-shooting, data logging, and code integration.

Augmented-reality system for demonstrations

The AR system described in Chapter 3 has enabled more communicative and experiential demonstrations in the Microdynamic Systems Lab. The lab very frequently demonstrates the ballbot and its capabilities to visiting scholars, and the AR system enables these expe-

riences to be more viewer-friendly.

Hardware updates for ballbot experiments

Over the course of this thesis work, multiple new systems have been added to the ballbot. These include turret and skin described in Chapter 3, the arm force sensors and end effectors, a new scanning laser range finder, the directional microphone, speakers, and point cloud sensor.

Software Implementation of Planning Architecture

All of the algorithms presented in this thesis have been implemented in C++ and python to run in real-time on the ballbot. This software is maintained in multiple git repositories on a lab-hosted GitLab. All together, numerous individuals, including multiple summer students and three other graduate students, have successfully interfaced with the system because of these updates. These updates also enabled the implementation of some of the presented work on a new ballbot with a different drive mechanism, the SIMbot [Seyfarth et al., 2016].

This thesis furthers the state of the art in pHRI, assistive robotics, and control of balancing mobile robots. By developing algorithms and testing experimentally on the ballbot, this work has demonstrated advances in forceful robotic interaction and navigation in dynamic environments.

The ballbot is unlike any mobile robot that came before it. It is the very first mobile robot to balance on a spherical wheel. This innovation yields the benefits of physical compliance and an advantageous form factor. The work of this thesis has utilized the benefits of the ballbot and explored additional interesting possibilities of navigation and forceful interaction. These new capabilities enable a vast range of future possibilities for the platform.

7.2 Future Work

The work of this thesis has enabled new capabilities for the ballbot in navigation and HRI and enabled exploration of new research areas, mostly in physical interaction with people. Some possible future directions for this line of research are presented in this section.

7.2.1 Assistance in Sit-to-Stand: User Subject Trial

The ballbot's ability to apply large forces and assist a person in standing from a chair was explored in Section 6.2. Although the data used to develop the controller were captured through a user study, the controller itself has not been used in a subject trial. Since the leading subject trial showed that the ballbot can safely be used for the physical HRI study, a future user study for helping people out of chairs can and should be run.

As with the leading study, the first test should be done using able-bodied participants. If this trial is successful, however, the capability of assisting in sit-to-stand would most benefit disabled and elderly people. As such, if IRB approval can be attained, a future study could investigate the use of the proposed controller in assisting individuals in these demographics in sit-to-stand. One interesting area of research is identifying whether unique strategies will be required for each of these populations to get out of a chair. The work presented in this thesis relies on two arms

holding the robot. Another strategy may use one arm to assist the subject while they use their other arm to balance using an armrest.

7.2.2 Crowd Navigation through Forceful Interaction

Section 6.4 presented a method for detecting collisions and using that information to modify a costmap. This costmap was used for planning and iteratively discovering obstacles in an environment. The method was developed with navigation through dense crowds in mind. A future area of research could deploy this system into dense crowds, such as those shown in Section 6.4.5. This study would assess the capabilities and limitations of this system. Ideally, this work could also be combined with other crowd navigation strategies, such as cooperative collision avoidance.

7.2.3 Leading by the Hand: Obstacles and Longer Trajectories

The subject trial presented in Section 6.3.4 assessed the capabilities of the ballbot in leading people between six goals in the Microdynamic Systems Lab. This involved navigation along straight-line paths. The underlying planning system, however, can generate trajectories through cluttered environments with obstacles. A future subject trial could investigate if this system is sufficient in physically leading people though a building. Such a study could also take advantage of the conversation system described in Section 6.3.6 and also could have subjects teach the ballbot locations through speech. The participants could then ask the ballbot to lead them to a location and assess how well the robot performed.

7.2.4 Leading by the Hand: Elderly and Disabled Individuals

As with assistance in sit-to-stand, the populations that could most benefit for assistive leading are the elderly and the disabled. A worthwhile future study could test the ballbot's ability to lead participants from said populations. It is possible that the methods and software that were shown to work with able-bodied people can also work for these groups. This, however, needs to be tested in a user study. Modifications such as lower speed and larger obstacle inflation could be necessary to accommodate a less capable participant.

Bibliography

Althoff, D., Wollherr, D., and Buss, M. (2011). Safety assessment of trajectories for navigation in uncertain and dynamic environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5407–5412.

BLS (2011). *Incidence rates for nonfatal occupational injuries and illnesses involving days away from work per 10,000 full-time workers by industry and selected events or exposures leading to injury or illness, private industry*. www.bls.gov/iif/oshwc/osh/case/ostb3210.pdf.

Cavens, D., Gloor, C., Illenberger, J., Lange, E., Nagel, K., and Schmid, W. A. (2007). Distributed intelligence in pedestrian simulations. In *Pedestrian and Evacuation Dynamics 2005*, pages 201–212. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chung, S. Y. and Huang, H. P. (2010). A mobile robot that understands pedestrian spatial behaviors. *International Conference on Intelligent Robots and Systems (IROS 2013)*, pages 5861–5866.

Crick, C., Jay, G., Osentoski, S., and Jenkins, O. C. (2012). ROS and Rosbridge. In *the seventh annual ACM/IEEE international conference*, pages 493–494, New York, New York, USA. ACM Press.

Dautenhahn, K., Walters, M., Woods, S., Koay, K. L., Nehaniv, C. L., Sisbot, A., Alami, R., and Siméon, T. (2006). How may I serve you? In *Proceeding of the 1st ACM SIGCHI/SIGART conference*, pages 172–179, New York, New York, USA. ACM Press.

Donald, B., Xavier, P., Canny, J., and Reif, J. (1993). Kinodynamic motion planning. *J. ACM*, 40(5):1048–1066.

Faiz, N., Agrawal, S. K., and Murray, R. M. (2001). Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities. *Journal of Guidance, Control, and Dynamics*, 24(2):219–227.

Flash, T. and Hogan, N. (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *The journal of Neuroscience*, 5(7):1688–1703.

Fliess, M., Lévine, J., Martin, P., and Rouchon, P. (1995). Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, 61(6):13–27.

Flores, M. E. and Milam, M. B. (2006). Trajectory generation for differentially flat systems via NURBS basis functions with obstacle avoidance. *2006 American Control Conference*, page 7 pp.

Haddadin, S. and Albu-Schaffer, A. (2008). Collision detection and reaction: A contribution

to safe physical human-robot interaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363. IEEE.

Hart, P., Nilsson, N., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Heerink, M., Kröse, B., Wielinga, B., and Evers, V. (2006). Human-robot user studies in eldercare: Lessons learned. *Smart Homes and Beyond*.

Helbing, D. and Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286.

Hollis, R. (2006). Ballbots. *Scientific American*, 18(1):58–63.

Hollis, R. L. (2010). *Dynamic balancing mobile robot*. US Patent Office.

Hung, H., Englebienne, G., and Cabrera Quiros, L. (2014). *Detecting conversing groups with a single worn accelerometer*. ACM, New York, New York, USA.

Interaction, O. N. (2010). *OpenNI User Guide*. OpenNI organization. Last viewed 19-01-2011 11:32.

Karamouzas, I., Heil, P., van Beek, P., and Overmars, M. H. (2009). A Predictive Collision Avoidance Model for Pedestrian Simulation. In *Motion in Games*, pages 41–52. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kerr, K. M., White, J. A., Barr, D. A., and Mollan, R. (2003). Analysis of the sit-stand-sit movement cycle in normal subjects. *Clinical Biomechanics*, pages 1–10.

Kohlbrecher, S., Von Stryk, O., and Meyer, J. (2011). A flexible and scalable slam system with full 3d motion estimation. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 155–160. IEEE.

Kuderer, M. and Kretzschmar, H. (2013). Teaching mobile robots to cooperatively navigate in populated environments. *International Conference on Intelligent Robots and Systems (IROS 2013)*, pages 3138–3143.

Kulyukin, V., Gharpure, C., Nicholson, J., and Pavithran, S. (2004). RFID in robot-assisted indoor navigation for the visually impaired. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, pages 1979–1984 vol.2. IEEE.

Lauwers, T., Kantor, G., and Hollis, R. (2005). One Is Enough! In *th International Symposium on Robotics Research*, pages 327–336, San Fransisco. Springer Berlin Heidelberg.

Lauwers, T. B., Kantor, G. A., and Hollis, R. L. (2006). A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2884–2889.

Lidoris, G., x0308, R., ller, F., Wollherr, D., and Buss, M. (2009). The Autonomous City Explorer (ACE) Project - Mobile Robot Navigation in Highly Populated Urban Environments. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1416–1422. IEEE.

Linder, T. and Arras, K. O. (2014). Multi-model hypothesis tracking of groups of people in RGB-

D data. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–7. IEEE.

Lu, D. V., Allan, D. B., and Smart, W. D. (2013). Tuning Cost Functions for Social Navigation. In *link.springer.com*, pages 442–451. Springer International Publishing, Cham.

Lu, D. V., Hershberger, D., and Smart, W. D. (2014). Layered costmaps for context-sensitive navigation. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 709–715. IEEE.

Luber, M., Stork, J., and Tipaldi, G. D. (2010). People tracking with human motion predictions from social forces. *Robotics and Automation*, pages 464–469.

Luca, A., Albu-Schaffer, A., Haddadin, S., and Hirzinger, G. (2006). Collision Detection and Safe Reaction with the DLR-III Lightweight Manipulator Arm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1623–1630. IEEE.

Magrini, E., Flacco, F., and De Luca, A. (2015). Control of Generalized Contact Motion and Force in Physical Human-Robot Interaction . In *IEEE International Conference on Robotics and Automation, 2015. Proceedings. ICRA '15*, Seattle, WA, USA.

Mampetta, A. K. (2006). Automatic transition of ballbot from statically stable state to dynamically stable state. Master's thesis, Carnegie Mellon University.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. *Robotics and Automation*, pages 2520–2525.

Mortl, A., Lawitzky, M., Kucukyilmaz, A., Sezgin, M., Basdogan, C., and Hirche, S. (2012). The role of roles: Physical cooperation between humans and robots. *The International Journal of Robotics Research*, 31(13):1656–1674.

Munro, B. J., Steele, J. R., Bashford, G. M., and Ryan, M. (1998). A kinematic and kinetic analysis of the sit-to-stand transfer using an ejector chair: implications for elderly rheumatoid arthritic patients. *Journal of Biomechanics*, 31:263–271.

Mutlu, B., Forlizzi, J., and Hodgins, J. (2006). A Storytelling Robot: Modeling and Evaluation of Human-like Gaze Behavior. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 518–523. IEEE.

Nagarajan, U. (2010). Dynamic Constraint-based Optimal Shape Trajectory Planner for Shape-Accelerated Underactuated Balancing Systems. *Robotics: Science and Systems*.

Nagarajan, U. (2012). *Fast and Graceful Balancing Mobile Robots*. PhD thesis, Carnegie Mellon University.

Nagarajan, U. and Hollis, R. (2013). Shape space planner for shape-accelerated balancing mobile robots. *The International Journal of Robotics Research*, 32(11):1323–1341.

Nagarajan, U., Kantor, G., and Hollis, R. (2012a). Integrated planning and control for graceful navigation of shape-accelerated underactuated balancing mobile robots. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 136–141.

Nagarajan, U., Kantor, G., and Hollis, R. (2014). The ballbot: An omnidirectional balancing mobile robot. *The International Journal of Robotics Research*, 33(6):917–930.

Nagarajan, U., Kantor, G., and Hollis, R. L. (2009a). Human-Robot Physical Interaction with dynamically stable mobile robots. In *Human-Robot Interaction (HRI), 2009 4th ACM/IEEE International Conference on*, pages 281–282.

Nagarajan, U., Kantor, G., and Hollis, R. L. (2009b). Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3743–3748. IEEE.

Nagarajan, U., Kim, B., and Hollis, R. (2012b). Planning in high-dimensional shape space for a single-wheeled balancing mobile robot with arms. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 130–135. IEEE.

Nourbakhsh, I. R., Bobenage, J., Grange, S., Lutz, R., Meyer, R., and Soto, A. (1999). An affective mobile robot educator with a full-time job. *Artificial Intelligence*, 114(1-2):95–124.

Paris, S., Pettré, J., and Donikian, S. (2007). Pedestrian Reactive Navigation for Crowd Simulation: a Predictive Approach. *Computer Graphics Forum*, 26(3):665–674.

Pettré, J., Ondřej, J., Olivier, A.-H., Cretual, A., and Donikian, S. (2009). Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *the 2009 ACM SIGGRAPH/Eurographics Symposium*, pages 189–198, New York, New York, USA. ACM Press.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source Robot Operating System. *ICRA workshop on open source software*, 3(3.2):5.

Richter, C., Bry, A., and Roy, N. (2013). Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. In *International Symposium on Robotics Research*, Singapore.

Rohmer, E., Singh, S. P. N., and Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1321–1326. IEEE.

Schearer, E. M. (2006). *Modeling dynamics and exploring control of a single-wheeled dynamically stable mobile robot with arms*. PhD thesis, Carnegie Mellon University.

Seyfarth, G., Bhatia, A., Sassnick, O., Shomin, M., Kumagai, M., and Hollis, R. (2016). Initial results for a ballbot driven with a spherical induction motor. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3771–3776. IEEE.

Shao, W. and Terzopoulos, D. (2007). Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274.

Shomin, M., Forlizzi, J., and Hollis, R. (2015). Sit-to-Stand Assistance with a Balancing Mobile Robot. In *2015 IEEE International Conference on Robotics and Automation*, Seattle, WA, USA.

Shomin, M. and Hollis, R. (2013). Differentially Flat Trajectory Generation for a Dynamically Stable Mobile Robot. In *2013 IEEE International Conference on Robotics and Automation*, pages 4452–4457, Karlsruhe, Germany.

Shomin, M. and Hollis, R. (2014). Fast, dynamic trajectory planning for a dynamically stable

mobile robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 3636–3641. IEEE.

Shomin, M., Vaidya, B., Hollis, R., and Forlizzi, J. (2014). Human-approaching trajectories for a person-sized balancing robot. *2014 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 20–25.

Shrestha, M., Schmitz, A., Uno, E., Yokoyama, Y., Yanagawa, H., Or, K., and Sugano, S. (2015). Using Contact-based Inducement for Efficient Navigation in Congested Environments . In *Planning, Control, and Sensing for Safe Human-Robot Interaction (ICRA 2015 Workshop)*.

Sibella, F., Galli, M., Romei, M., Montesano, A., and Crivellini, M. (2003). Biomechanical analysis of sit-to-stand movement in normal and obese subjects. *Clinical Biomechanics*, 18(8):745–750.

Sira-Ramirez, H. and Agrawal, S. K. (2004). *Differentially flat systems*. CRC Press.

Spinello, L. and Siegwart, R. (2008). Human detection using multimodal and multidimensional features. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3264–3269. IEEE.

Sup, F., Bohara, A., and Goldfarb, M. (2008). Design and Control of a Powered Transfemoral Prosthesis. *The International Journal of Robotics Research*, 27(2):263–273.

Thompson, S., Horiuchi, T., and Kagami, S. (2009). *A probabilistic model of human motion and navigation intent for mobile robot path planning*. IEEE.

Trautman, P., Ma, J., Murray, R. M., and Krause, A. (2015). Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356.

Triebel, R., Arras, K., Alami, R., Beyer, L., Breuers, S., Chatila, R., Chetouani, M., Cremers, D., Evers, V., Fiore, M., Hung, H., Ramirez, O. A. I., Joosse, M., Khambhaita, H., Kucner, T., Leibe, B., Lilienthal, A. J., Linder, T., Lohse, M., Magnusson, M., Okal, B., Palmieri, L., Rafi, U., van Rooij, M., and Zhang, L. (2015). SPENCER: A Socially Aware Service Robot for Passenger Guidance and Help in Busy Airports . In *Field and Service Robots*, Toronto.

Van Nieuwstadt, M. and Murray, R. (1998). Real Time Trajectory Generation for Differentially Flat Systems. *International Journal of Robust and Nonlinear Control*, pages 995–1020.

Vasquez, D., Okal, B., and Arras, K. O. (2014). Inverse Reinforcement Learning algorithms and features for robot navigation in crowds: An experimental comparison. *Intelligent Robots and Systems (IROS 2014)*, pages 1341–1346.

Webb, D. J. and van den Berg, J. (2013). Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5054–5061.