Fast and Graceful Balancing Mobile Robots

Umashankar Nagarajan

CMU-RI-TR-12-16

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

July 2012

Thesis Committee
Ralph Hollis, Chair
George Kantor
Howie Choset
Russ Tedrake, MIT

Copyright © 2012 Umashankar Nagarajan. All rights reserved.

Keywords: Balancing Mobile Robots, Underactuated Systems, Trajectory Planning, Shape Space Planning, Integrated Motion Planning and Control

To my parents
Nirmala Nagarajan and Nagarajan Rangasamy

Abstract

Personal mobile robots will soon be operating and closely interacting with us in human environments. Balancing mobile robots can be effective personal robots as they can be tall enough for eye-level interaction and narrow enough to navigate cluttered environments, and they also have the dynamic capabilities to move with speed and grace comparable to that of humans. The work presented in this thesis enables balancing mobile robots to achieve fast and graceful navigation in human environments while handling disturbances and dynamic obstacles. This work particularly focuses on the ballbot, a human-sized mobile robot that balances on a single ball.

The natural dynamics of balancing mobile robots have to be exploited in order to make them achieve fast and graceful motions. This thesis introduces *shape-accelerated balancing systems* as a special class of underactuated systems to which balancing mobile robots like the ballbot belong. They have a special property wherein non-zero shape configurations result in accelerations in the position space. This thesis presents a trajectory planning algorithm that plans shape trajectories for shape-accelerated balancing systems, which when tracked will result in optimal tracking of desired position trajectories. It also presents experimental results of the ballbot with arms successfully achieving desired position space motions using body lean motions, arm motions, and combinations of the two, and also handle cases where the arms are artificially constrained.

This thesis presents an integrated motion planning and control framework based on sequential composition, which enables balancing mobile robots to achieve graceful navigation. It presents controllers called *motion policies* that are designed to achieve fast, graceful motions in small domains of the position space that are collision-free. It introduces the *gracefully prepares relationship* that guarantees a valid sequential composition of motion policies to produce overall graceful motion. It presents an automatic instantiation procedure that deploys these motion policies to fill a map of the environment, and also a motion planner that plans in the space of these gracefully composable motion policies to achieve desired navigation tasks. This thesis also presents experimental results of the ballbot successfully achieving different navigation tasks while handling disturbances and dynamic obstacles.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Ralph Hollis, for all his support and guidance throughout my graduate life. I have learnt a lot from him, especially about designing and building robots, and also about presenting research. I also thank him for all the ohiopyle bike trips, ski trips, and factory visits. I owe special thanks to his wife, Elizabeth Hollis, for her love and affection, and also for the tastiest banana nut breads and chocolate-chip cookies I have ever had.

I would like to thank my committee members, George Kantor and Howie Choset, for being in all my committees, and for all their valuable comments and discussions throughout my graduate life. I really appreciate their time and effort in helping me do my best. I would also like to thank my external committee member, Russ Tedrake, for his useful comments and discussions.

I would like to thank all my friends at Carnegie Mellon University for making my graduate life both a fun and a learning experience. I would like to especially thank Brian Becker, Michael Furlong, Siddharth Sanan, Joydeep Biswas, Santosh Divvala, Brina Goyette, Prasanna Velagapudi, Pyry Matikainen, Heather Jones, Heather Justice and Nathan Brooks for helping me remain sane. The dinner trains, coffee times, eat'n park science sessions, pamela's brunches, movie nights, and racquetball sessions are fond memories of my graduate life.

I would like to thank my lab mates Bertram Unger and Hanns Tappeiner for the many useful conversations we have had. I would also like to thank Jun Xian Leong, Byungjun Kim, Michael Shomin and Sudhir Katta for their great help in conducting the ballbot experiments.

I would like to thank my wife, Shirpaa Manoharan, for all her love and support, and also for making my life so much better. Finally, I would like to thank my parents, Nirmala Nagarajan and Nagarajan Rangasamy, who have always trusted me and supported me in everything I have done. I would like to dedicate my PhD to my parents. Without their love and support, this degree would not have been possible.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Though the field of robotics has grown significantly over the last few decades, it is still in its infancy when it comes to personal robotics. As of today, personal robots are restricted to science fiction movies. There are a lot of challenges in the development of personal robots in all facets of their operation like perception, intelligence, navigation and interaction. The work presented in this thesis addresses some of the challenges in navigation for robots operating in human environments. In particular, this work focuses on developing balancing mobile robots that navigate human environments with speed and grace comparable to that of humans.

1.1 Motivation

Personal robots operating in human environments and interacting with humans do not have to necessarily look like humans but must move, act and interact like humans. Therefore, personal robots must be of human size, in both height and footprint. They must be tall enough for eye-level interaction and narrow enough to navigate cluttered spaces.

Traditionally, the robotic locomotion platforms are three or four-wheeled platforms that are statically stable, $i.e.$, they stand still when powered down. However, human-sized statically stable mobile robots, shown in Fig. 1.1, need wide bases to have large polygons of support, and a lot of dead weight in their bases to keep their centers of gravity as low as possible. A high center of gravity and/or a small polygon of support will cause a statically stable mobile robot to tip over easily, which is definitely undesirable. Moreover, their chances of tipping over drastically increase when lifting heavy objects as the net center of gravity can shift outside the polygon of support as shown in Fig. 1.2(a). This tipping behavior can also be caused by large

acceleration or deceleration, and moving up or down steep slopes as shown in Fig. 1.2(b). The wide bases provide large polygons of support, but make statically stable mobile robots unsuitable for operation in human environments that are often narrow and cluttered.

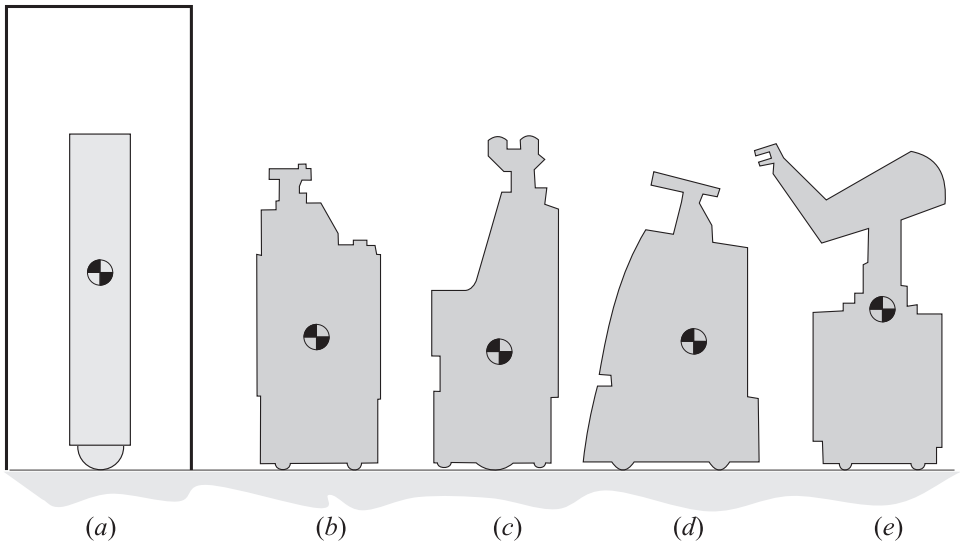


Figure 1.1: Approximate sketches of the ballbot and some statically stable mobile robots: (a) The ballbot [41], (b) Xavier [116], (c) Nursebot [4], (d) Minerva [127], (e) Juliet [48]. The rectangle around the ballbot represents a standard house doorway [60]. (*Courtesy:* Ralph Hollis; Appeared in [60])

The above mentioned drawbacks of statically stable robots can be avoided by building mobile robots that actively balance, just like humans do. Unlike statically stable mobile robots, balancing mobile robots are dynamically stable, and can be tall and skinny with high centers of gravity. They can have small footprints as they are continually balancing, and can accelerate or decelerate quickly [41]. Balancing mobile robots can also avoid tipping by actively compensating for the shift in the center of gravity, and balance accordingly. For the tipping scenarios presented in Fig. 1.2, it can be seen that a balancing mobile robot can maintain balance by leaning back while carrying a heavy object, and by leaning into the ramp while ascending/descending a steep slope.

Moreover, balancing mobile robots are physically interactive. One can push, nudge and kick a statically stable robot, but it is not going to respond. Humans have the need to physically interact with machines in their environments, especially when they are their personal robotic assistants. Balancing mobile robots are naturally reactive as they inherently detect physical interactions as disturbances to their balancing behavior. This makes balancing mobile robots responsive to

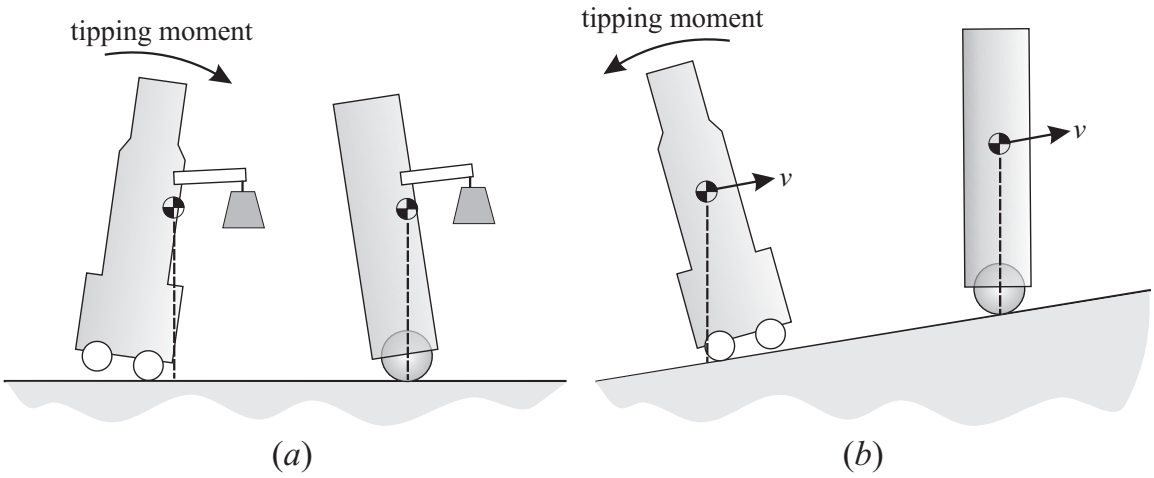
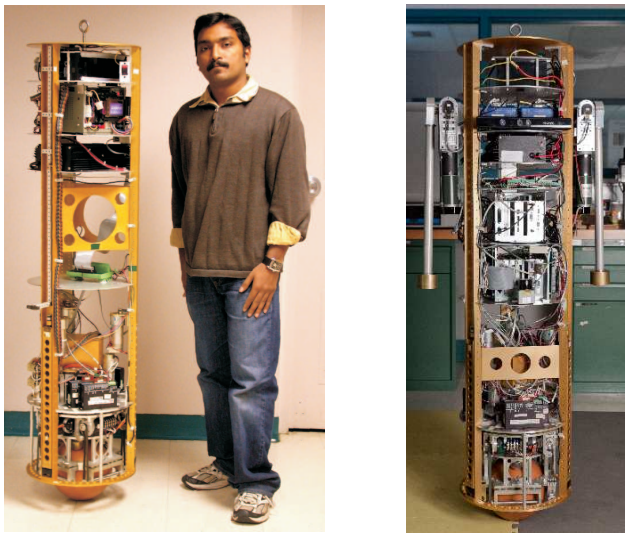


Figure 1.2: (a) A statically stable robot can tip over when attempting to lift a heavy weight, whereas a balancing robot can lean to keep the total center of mass over the point of ground support; (b) A statically stable robot could tip when going up or down slopes, whereas a balancing robot can stay balanced on slopes (*Courtesy:* Ralph Hollis; Appeared in [79]).

human touch [75]. Moreover, balancing mobile robots can also be effective mobile manipulators [13] with the ability to maintain postural stability, generate forces on external objects, and withstand greater impact forces. All these characteristics make balancing mobile robots ideal candidates for personal robotic assistants that move, act and interact like humans.

Balancing mobile robots include two-wheeled mobile robots like the Segway [83], one-wheeled mobile robots like the ballbot [41], and legged robots such as BigDog [97] and MABEL [31]. The continuous dynamics of all these robots can be represented using simple wheeled inverted pendulum models, and hence any navigation procedure developed for the simplified models can be extended to these robots. The work presented in this thesis focuses on simple, balancing wheeled locomotion platforms that can be used as personal mobile robots in human environments. In particular, it focuses on the ballbot [41], a human-sized mobile robot that balances on a single ball as shown in Fig. 1.3(a). It is a 3D omni-directional wheeled inverted pendulum robot with a cylindrical body atop a ball. The ball is fully actuated, whereas the body is not. The principle used to balance the ballbot is same as that used for balancing a stick, $i.e.$, if the robot leans, a balancing controller rolls the ball in the direction of the lean in order to keep the robot upright. The ballbot has an inertial measurement unit (IMU), which provides the robot's body lean with respect to gravity, and the ball is driven using the inverse of an old-fashioned mouse-ball drive. Recently, a pair of two degrees of freedom (DOF) arms was added to the

ballbot as shown in Fig. 1.3(b). A detailed description of the ballbot's hardware is presented in Chapter 3. The ballbot's dynamic stability enables it to achieve fast, dynamic and graceful motions, while the ball enables it to achieve omnidirectional motion, which is a key feature for operation in cluttered human environments.



(a) (b)

Figure 1.3: (a) The ballbot balancing (*Courtesy:* Ralph Hollis; Appeared in [75, 77, 78, 79, 80]), and (b) the ballbot with a pair of 2-DOF arms (*Courtesy:* Michael Shomin; Appeared in [73]).

1.2 Thesis Objective

As discussed in Sec. 1.1, balancing mobile robots like the ballbot have the dynamic capabilities to navigate human environments with speed and grace comparable to that of humans, and this thesis presents the work done in realizing such motions in a controlled fashion.

The objective of the work presented in this thesis is to *enable balancing mobile robots like the ballbot to achieve fast and graceful navigation in human environments*. In an attempt at realizing the dream of placing fast, graceful balancing mobile robots as personal robots in human environments, this work answers two major research questions listed below.

RQ 1: *How to exploit the natural dynamics of balancing mobile robots to produce fast and dynamic motions?*

RQ 2: *How to develop a planning and control framework that will enable graceful navigation of balancing mobile robots in an obstacle-ridden environment while handling disturbances?*

1.3 Challenges

The objective of the work presented in this thesis is to enable balancing mobile robots like the ballbot to achieve fast and graceful navigation in human environments. This section briefly describes the challenges faced in achieving this objective.

Balancing (dynamically stable) mobile robots are underactuated mechanical systems, $i.e.$, systems with fewer independent control inputs than the degrees of freedom [118]. An interesting and troubling factor in planning and control of such underactuated systems is the constraint on their dynamics by virtue of underactuation. These constraints are second-order nonholonomic [99] constraints, $i.e.$, non-integrable acceleration/dynamic constraints. These constraints restrict the family of trajectories that the configurations can follow. Balancing mobile robots are destabilized by gravitational forces, and hence have to maintain balance while trying to track arbitrary trajectories. The underactuation and the resulting unstable dynamics make the navigation a challenging planning and control task. For example, in the case of the ballbot shown in Fig. 1.3(a), the ball is directly actuated, whereas the body is not. Moreover, there is a strong coupling between the dynamics of the ball and the body. If the ball is rolled, the body falls; and if the body is held upright by moving the ball, then the ball is not in its desired position on the floor. Hence, achieving desired fast, dynamic motions for balancing mobile robots like the ballbot while maintaining balance is a challenging task.

Traditionally, motion planning and control for mobile robots have been decoupled. On one hand, the motion planner takes into account the obstacles in the environment and also the workspace constraints, but does not account for the dynamics of the system and the constraints on them. It also does not have any knowledge of the limitations of the controller used to achieve the generated motion plans. On the other hand, the controller does not have any knowledge of the workspace constraints, the obstacles in the environment or the navigation task it is trying to achieve. Though it is possible to make dynamic, underactuated balancing systems navigate environments using these decoupled procedures, they are often sub-optimal and result in jerky motions that are ungraceful. Moreover, when disturbed, these procedures often either result in collisions with obstacles or drive the system unstable. Therefore, in order to achieve robust, fast, graceful and collision-free motions for balancing mobile robots like the ballbot, the motion planning and control should be integrated such that both the motion planner and the controller understand the dynamics of the system, the constraints on them, and each other's details.

1.4 The Need for Graceful Robot Motion

Balancing mobile robots have the ability to move with speed and grace comparable to that of humans. The objective of the work presented in this thesis is to enable balancing mobile robots like the ballbot to achieve graceful navigation in human environments. This section presents the definition of a graceful robot motion used in this work, and also highlights its importance.

A dictionary defines "a graceful motion to be one that is seemingly effortless and natural". Gulati and Kuipers [32] define graceful motion for an intelligent wheelchair to be a fast, safe, comfortable and intuitive motion. A comfortable motion is defined as one whose velocity, acceleration trajectories are continuous and bounded with low jerk. The work presented in this thesis uses a similar definition for graceful motion: *Any feasible robot motion is defined to be graceful if its configuration variables' position, velocity and acceleration trajectories are continuous and bounded*.

Apart from being visually appealing, a graceful robot motion has a variety of advantages. Continuous and bounded acceleration trajectories result in low jerk. Graceful, low jerk motions result in smoothed actuator loads [57] because jerk is the rate of change of acceleration, and it is directly proportional to the torque rate of actuators. High jerk motions, on the other hand, can excite resonant frequencies of the robot, which can drive a balancing system unstable. In order to better understand the extent to which jerk and torque rate happens, let's simulate and compare two cases where the ballbot [61] attempts to track a desired 2 m motion composed of two trajectories, one with a continuous acceleration trajectory and the other with a discontinuous acceleration trajectory. The resulting jerk and torque rate trajectories from tracking these motions are shown in Fig. 1.4. The maximum jerk in the discontinuous acceleration case (52.73 m/s^3) is 179 times more than that in the continuous acceleration case (0.29 m/s^3), while the maximum torque rate in the discontinuous acceleration case (117.07 Nm/s) is 15.25 times more than that in the continuous acceleration case (7.68 Nm/s).

It has been shown in biomechanics literature that humans tend to move such that their movements minimize jerk over their entire motion [40]. Minimum-jerk models have been proposed to model arm movements [19], and are used in various rehabilitation and haptic applications [2]. Smoothness is a characteristic of unimpaired human movements, and humans generally associate "high jerk" motions to "panic" motions [2, 103]. Therefore, humans are unlikely to feel comfortable around robots that have high jerk, non-smooth (ungraceful) motions.

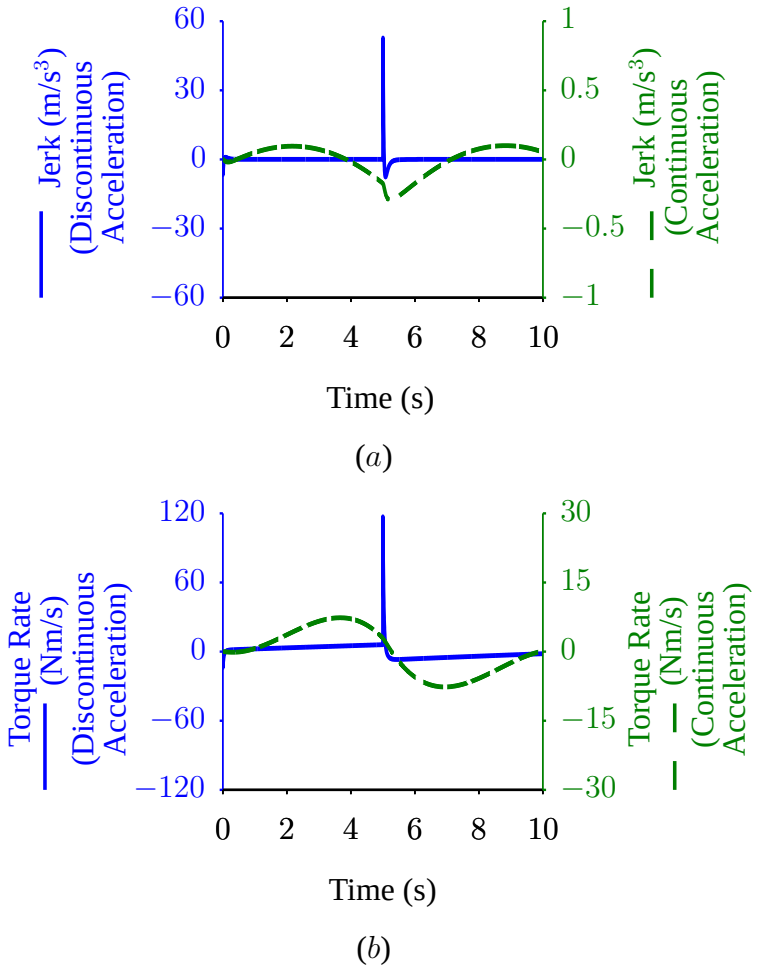


Figure 1.4: Comparing the results of tracking discontinuous and continuous acceleration trajectories in simulation: (a) Resulting jerk trajectory; (b) Resulting torque rate trajectory.

Moreover, personal robots in human environments are likely to engage in physical interactions with humans, wherein high jerk motions can be undesirable as humans interacting with the robots will experience the same. Certain physical interactions may be more sensitive than others, for example, guiding a visually-impaired person by hand and helping an elderly person get up from a chair undoubtedly require graceful, low jerk motions. Other tasks like carrying an open container with fluid demand low jerk motions as fluid spillage is again undesirable. Therefore, in order to build successful personal robots that operate and interact in human environments, it is important to ensure that they have graceful motions.

1.5 Approach

This section briefly describes the approaches used in this work to answer the research questions listed in Sec. 1.2.

1.5.1 Planning in Shape Space

This thesis addresses the question of achieving fast, dynamic motions for balancing mobile robots (RQ 1) by presenting trajectory planning algorithms that plan in the shape space of the system to achieve desired fast, dynamic motions in the position space. The configuration space of any dynamic system can be split into *position space* and *shape space*. Position variables represent the position of the robot in the world frame, and the dynamics of mobile robots are independent of transformations of their position configurations. However, shape variables are those that affect the inertia matrix of the system, and dominate the system dynamics. There is a strong coupling between the dynamics of shape and position configurations in balancing mobile robots like the ballbot, and this thesis presents procedures that exploit this inherent dynamic coupling to achieve fast, dynamic motions.

Although navigation tasks are generally posed as desired motions in the position space without any specifications on shape space motions, motions in the shape space cannot be ignored for balancing mobile robots like the ballbot. Since the shape dynamics dominate the system dynamics, any desired motion in the position space can be successfully achieved only if an appropriate motion in the shape space is planned and tracked. For example, the ballbot [76] cannot track fast position trajectories while maintaining an upright position, $i.e.$, zero lean angle (no shape change) because of the dynamic coupling between the motion of the ball and the body. Any control attempt to do so will result in jerky motions or drive the system unstable. In order to achieve fast and dynamic motions, the ballbot must lean. Section 4.2 presents planning algorithms that generate shape trajectories ($e.g.$, lean angle trajectories for the ballbot) using just the dynamic constraint equations for balancing mobile robots, which when tracked will result in optimal tracking of desired position trajectories.

1.5.2 Graceful Navigation

The shape trajectory planning algorithms presented in Section 4.2 enable balancing mobile robots like the ballbot to achieve fast, dynamic motions, but the motions are not scalable for navigation

purposes. Moreover, since these motions are achieved by tracking trajectories, they are not good at handling large disturbances. Large disturbances can either make a balancing system unstable or result in collisions with obstacles in the environment.

This thesis presents an integrated motion planning and control framework to achieve fast and graceful navigation for balancing mobile robots in an obstacle-ridden environment while handling disturbances and dynamic obstacles (RQ 2). This integrated motion planning and control framework uses controllers called *motion policies* that result in fast, graceful motions in small, collision-free domains of the position space. Each motion policy consists of: (i) a motion primitive, $i.e.$, feasible state trajectories that result in fast, graceful motions in the position space; (ii) a feedback control law that tracks the motion primitive; and (iii) a domain for its control law that is collision-free.

Unlike traditional motion planners that plan in the space of cells or paths, the motion planner presented in this thesis plans in the space of motion policies, $i.e.$, controllers. The motion planner chooses a sequence of gracefully composable motion policies to achieve the overall navigation task. Local, valid motion policies that result in fast, graceful and simple motions in small domains of the position space are sequentially composed to produce a global motion policy that results in a fast, graceful and complicated motion in the position space. This procedure ensures that the high-level motion planner has complete knowledge of the low-level controller it uses, and the low-level controller has knowledge of the environment and the navigation task it achieves, thereby forming a truly integrated motion planning and control framework that enables balancing mobile robots like the ballbot to achieve graceful navigation in human environments.

1.6 Outline

Chapter 2 presents the related work for this thesis. Chapter 3 presents the system description, modeling, control architecture and capabilities of the ballbot. Chapter 4 introduces shape-accelerated balancing systems, a special class of underactuated systems to which balancing mobile robots like the ballbot belong. It then presents an optimal shape trajectory planner that plans motions in the shape space, which when tracked will result in optimal tracking of desired motions in the position space. Chapter 5 presents a procedure to design gracefully composable motion policies, and also presents a motion planner that plans in the space of gracefully composable motion policies to achieve desired navigation tasks in the presence of static and dynamic obstacles. Figure 1.5 depicts a high-level overview of the work presented in this thesis. Chapter 6 presents

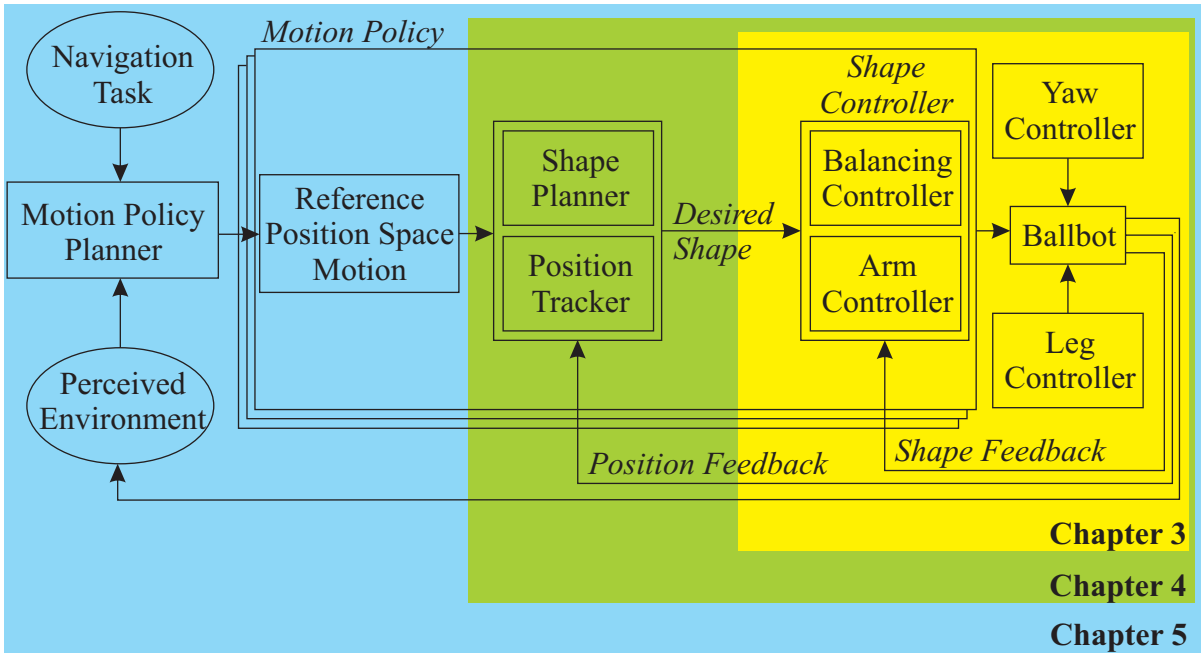


Figure 1.5: A high-level overview of the work presented in this thesis.

the conclusions and contributions of this thesis, and also discusses some future directions of research.

Chapter 2

Related Work

2.1 Balancing Mobile Robots

There has been a significant growth of interest in developing balancing mobile robots in the last decade. Two-wheeled balancing mobile robots ([33], [34], [123], and [12]) became popular after the introduction of the *Segway Robotic Mobility Platform* [83]. Rod Grupen and his group introduced a two-wheeled dynamically stable mobile robot called *uBot* [12], which is used as a mobile manipulation research platform [14]. They showed that balancing robots can be effective mobile manipulators [12] with the ability to maintain postural stability, generate forces on external objects, and withstand greater impact forces. Dean Kamen introduced *iBot* [43], a balancing wheelchair, and demonstrated the advantages of balancing wheelchairs over the traditional statically stable ones. Anybots [3] introduced a tele-presence robot that balances on two wheels. Mike Stilman and his group introduced *Golem Krang* [122], a two-wheeled balancing mobile manipulator platform that has the capability of autonomously standing and sitting. They showed successful control strategies for two-wheeled balancing robots that avoid low ceilings and other vertical obstacles [126].

Our group introduced the ballbot [61], an omni-directional dynamically stable mobile robot whose single-wheel design circumvents the limitations associated with kinematic constraints of two-wheeled robots. Recently, several other groups have begun exploring single-wheel designs [38, 55, 100]. Masaaki Kumagai developed the *Ball-IP* [55], a ball balancing robot, and demonstrated that it can be used in cooperative transportation of wooden frames [56]. A group of mechanical engineering students at ETH Zurich developed the *Rezero* [100], and re-emphasized the dynamic capabilities of ball balancing mobile robots.

2.2 Underactuated Systems

Balancing mobile robots are members of a class of underactuated systems [118], systems with fewer independent control inputs than the number of degrees of freedom. Trajectory planning and control of underactuated mechanical systems has attracted growing attention over the years. There is a large body of literature on trajectory planning for nonholonomic systems with kinematic constraints, ranging from theoretical foundations [58] to practical implementations such as multi-wheeled mobile vehicles [42, 59, 117]. Underactuated systems with dynamic constraints have been approached from the controls perspective (*e.g.*, acrobot swing-up [120]) as well as from the planning perspective (*e.g.*, airship path planning [49]). A detailed analysis of underactuated manipulators (with passive joints) from both the dynamic and control point of view was presented in [87]. Rathinam and Murray developed methods to determine configuration flatness of Lagrangian control systems underactuated by one control [98].

Nonlinear control procedures [44, 45] for regulation of underactuated mechanical systems based on partial feedback linearization were introduced in [118]. In [86], Olfati-Saber introduced explicit cascade normal forms for underactuated mechanical systems with two degrees of freedom and kinetic symmetry. He also presented different classes of high-order underactuated mechanical systems and partial feedback linearization [119] techniques for reduction and control. Underactuated balancing systems have unstable zero dynamics, and are called nonminimum-phase systems. Accurate tracking of arbitrary configuration trajectories for such systems is not possible. A variety of nonlinear inversion based approaches [15, 16] have been used in literature to achieve approximate tracking of desired trajectories for such systems. One such dynamic inversion method was developed by Neil Getz [29]. Getz and Hedrick [27] developed a nonlinear controller based on internal equilibrium manifold for nonlinear nonminimum-phase systems that provided a larger region of attraction over linear regulators, and enabled better output tracking while maintaining balance [28]. He demonstrated these control procedures on bicycle models [26], which were extended by Yi *et al.* to motorcycle models [133, 134]. These dynamic inversion based approaches are computationally expensive, and cannot be run real-time on robots. This thesis presents trajectory planning algorithms that are fast enough to run real-time on robots. Shiriaev *et al.* [113] presented a constructive tool for generation and orbital stabilization of periodic solutions for underactuated nonlinear systems with a single passive degree of freedom. They achieved this using virtual holonomic constraints and transverse linearization [114]. In [115], they developed techniques for transverse linearization and orbital stabilization of periodic

motions for underactuated systems with arbitrary number of passive degrees of freedom. One of the objectives of the work presented in this thesis is to find functions similar to these virtual holonomic constraints, which map desired motions in the position space to motions in the shape space.

2.3 Planning in Shape Space

Ostrowski [89] presented a formulation for undulatory robotic locomotion [91] in mechanical systems with nonholonomic constraints and symmetries [92]. Geometric mechanics tools were used to study the effect of internal shape changes on net changes in position and orientation. He presented mechanical connection and reconstruction equation [7] that relate shape changes to momentum and position [90]. He presented various gaits for snakeboards [64] and Hirose snakes [39], and addressed their controllability issues [90]. However, planning procedures that plan for motions in the shape space to achieve desired motions in the position space were not presented.

Shammas *et al.* presented a variety of gait design tools for principally kinematic, purely mechanical systems [108, 111] and dynamic systems with nonholonomic velocity constraints [109, 110, 112]. They presented tools for generation of both kinematic and dynamic gaits, unlike just kinematic gait generation techniques introduced in [9] for snakeboards. The gaits were defined as closed-loop motions in internal shape variables, which produced desired position changes in the body coordinate frame, and they were generated using *height functions* [112]. However, these design tools are not applicable to dynamic systems with nonholonomic acceleration constraints. Hatton and Choset [37] used the connection, which relates the body velocity to internal shape changes, to create a set of vector fields on the shape space called *connection vector fields*. Each connection vector field corresponds to one component of the body velocity, and informs how a given shape change will move the system through its position space. The main advantage of this approach is that it is not restricted to gaits, and can be used for any general shape change. However, this procedure was restricted to principally kinematic and purely mechanical systems [108, 111].

2.4 Hybrid Control

Hybrid control approaches have been primarily used for feedback stabilization of underactuated systems [1, 135]. Sanfelice and his group [84, 105] presented a “throw-and-catch” hybrid control

strategy for robust global stabilization of pendubot. Their control strategy combined local feedback stabilizers and open-loop controls to steer state trajectories towards desired points in state space. They also used a *bootstrap* feedback controller that is capable of steering state trajectories to a neighborhood of states where feedback stabilizers or open-loop controls can be used.

The work presented in this thesis uses a hybrid control architecture for motion planning. In the last decade, there has been a large body of work on using hybrid control techniques for motion planning that will avoid decoupling between planners and controllers. Burridge *et al.* [10] introduced *Sequential Composition*, a controller composition technique that connects a palette of controllers, and automatically switches between them to generate a globally convergent feedback policy. They showed that the stability of individual control policies guarantee the stability of the overall hybrid policy, and showed results on a robot juggling a ping-pong ball with a paddle. The robustness of this approach to perturbations was also demonstrated. However, they used a manual sequence of policies, and did not present any planning procedure for obtaining such a sequence. Moreover, they used a palette of control policies that contained only convergent policies.

Sequential composition was successfully applied to a variety of systems [46, 50, 96, 102]. In [101], Rizzi used sequential composition to simplify motion programming for an idealized holonomic second-order dynamical robot. Quaid and Rizzi [96] extended sequential composition to planar motors with velocity and acceleration bounds. In the control of wheeled mobile robots, Kantor and Rizzi [46] used sequential composition to navigate a kinematic unicycle robot using visual servoing control policies with limited field of view. A variable constraint controller was used to define individual control policies. Patel *et al.* [93] used sequential composition to define switching policies for a nonholonomic wheelchair that navigates through doorways using visual servoing with limited field of view. In both [46] and [93], the defined control policies included both convergent control policies and control policies that had exit velocities. Conner *et al.* [11] called control policies with exit velocities as *flow-through* policies.

Conner *et al.* [11] extended sequential composition to produce an integrated motion planning and control procedure to achieve global navigation objectives for convex-bodied wheeled mobile robots navigating amongst static obstacles. They also presented a sampling-based approach for partially automating the policy deployment process. Conner *et al.* [11] primarily dealt with kinematic wheeled mobile robots with non-circular shapes, and did not deal with systems having more complicated dynamics like balancing mobile robots. The control policy domains were restricted to the configuration space of the system, and the policy deployment process was not

fully automated. Moreover, their integrated motion planning and control procedure did not guarantee overall graceful motion. The work presented in this thesis extends this integrated motion planning and control framework to balancing mobile robots, and also guarantees overall graceful motion.

Belta *et al.* [5] presented a hybrid control policy that used piecewise affine control policies defined over simplices. This approach involved motion planning in the space of simplices, and the control policies were defined over simplices in order to induce desired closed-loop motions. These methods were developed for fully actuated kinematic robots with velocity bounds, and underactuated kinematic unicycles with forward and turning speed bounds. These methods were not presented for systems with significant dynamics like balancing mobile robots. Moreover, this procedure did not guarantee overall graceful motion.

Manikonda *et al.* [67, 68] introduced *MDLe*, as an extension to motion description language (MDL) presented in [8]. The robot behaviors were formalized in terms of kinetic state machines, a motion description language, and the interaction of kinetic state machines with real-time information from limited-range sensors. They demonstrated MDLe in the area of motion planning for nonholonomic kinematic unicycles. Sensor based triggers were used to avoid collisions and to switch behaviors. They used potential function based local planners to plan collision-free paths assuming a holonomic robot, and then generated feasible paths that obeyed the configuration constraints. This provided the sequence of control points to which the robot had to be steered and then, behaviors that steered the robot to these desired control points were selected. This approach was still a decoupled one with no integration of planning and control. Moreover, they used only open-loop controls, and did not account for the domains of controllers.

Marigo and Bicchi [69] developed *Control Quanta* as a motion planning method for driftless systems with symmetries. Each control quantum was defined as an open-loop control trajectory with zero control inputs at the start and the end, and therefore, it only resulted in a rest-to-rest motion of the system. Kovar *et al.* [54] introduced *Motion Graphs* to build complex motions of animated figures from motion-capture sequences. The motion-capture sequences formed motion primitives, and given a finite number of motion primitives, a motion graph was constructed as a directed graph representing rules of their valid sequential composition. They defined two motion primitives to be sequentially composable if one primitive's end state was "close enough" to the start state of the other. Therefore, two motion primitives can be sequentially composed as long as discontinuities in their state trajectories are not perceivable by the user. However, for purposes of robot navigation, this approach results in ungraceful motion of the system.

Frazzoli *et al.* [20, 21, 24] introduced *Maneuver Automata* as a generalization of control quanta [69] and motion graphs [54]. They used open-loop maneuvers and steady state (trim) trajectories as motion primitives, and these motion primitives were concatenated based on predefined relationships. They used algorithms based on Rapidly-exploring Random Trees (RRT) [62] for motion planning in maneuver space, and demonstrated aggressive maneuvering capabilities of autonomous helicopters in simulation [23, 25]. The algorithm presented did not deal with coverage but rather, stopped when a sequence of motion primitives to the goal was found. Therefore, every time the state exited the defined domain, the algorithm had to replan. They also presented *Robust Hybrid Automata* [20, 22] that used closed-loop control for its maneuvers. Although this approach ensured overall stability of the closed-loop system while switching between motion primitives, it did not ensure closed-loop graceful motion.

Russ Tedrake [124, 125] introduced *LQR-trees*, a feedback motion planning algorithm that combines locally valid linear quadratic regulator (LQR) controllers into a nonlinear feedback policy that globally stabilizes a goal in state space. LQR-trees consist of a sparse set of feasible trajectories to the goal stabilized by LQR controllers along with conservative estimates of their basins of attraction. The algorithm probabilistically covers the bounded state space with such basins and ensures that the initial conditions capable of reaching the goal will stabilize to the goal. The estimation and verification of stability regions are computationally expensive, and hence do not allow real-time planning. However, the approaches presented by his group [125, 128] to estimate invariant domains for control policies can be used in the design of motion policies presented in this thesis. Recently, his group presented an offline approach to design a library of parameterized feedback controllers that can be used for real-time motion planning [65].

Chapter 3

The ballbot

The ballbot is an underactuated, dynamically stable mobile robot that balances on a single ball. It is a 3D wheeled inverted pendulum robot that is capable of omnidirectional motion.

3.1 History

In 2005, Ralph Hollis from the Microdynamic Systems Laboratory at Carnegie Mellon University, Pittsburgh, USA built the first successful ballbot [61] shown in Fig. 3.1. It was intentionally built to be of human-size with the vision of developing tall and skinny robots that navigate human environments. A general discussion on the capabilities and advantages of such balancing robots was presented in [41]. Tom Lauwers and George Kantor were the first to make the ballbot successfully balance and stationkeep [60, 61]. The ballbot uses a triad of legs to gain static stability when powered down as shown in Fig. 3.1(c). In 2006, Anish Mampetta [66] explored different approaches to enable the ballbot to automatically transition from this statically stable state (Fig. 3.1(c)) to the dynamically stable, balancing state (Fig. 3.1(b)). In the same year, Eric Schearer [107] explored the dynamic effects of adding arms to the ballbot in simulation. He also developed unified arm, balancing and stationkeeping controllers in simulation. Kathryn Rivard, Suresh Nidhiry and Kalicharan Karthikeyan played a significant role in the development of the hardware drivers and the initial simulation software for the ballbot. Until 2007, the ballbot was able to balance, but didn't do much else.

This thesis presents the work done with the ballbot since 2007. The first contribution of the work presented in this thesis is to have enabled the ballbot to balance reliably, to be robust to disturbances, and also to be physically interactive.

(*a*) (*b*) (*c*)

Figure 3.1: The ballbot: (a) CAD drawing with its principal components marked; (b) balancing; and (b) statically stable with the legs down. (*Courtesy:* Ralph Hollis; Appeared in [75, 80])

3.2 System Description

This section presents a detailed description of the ballbot's hardware designed and built by Ralph Hollis. The work presented in this thesis does not involve the design of any of the hardware components described below, but involves the design of controllers for these components, which are described in Sec. 3.5.

The ballbot shown in Fig. 3.1 consists of a cylindrical body on top of a ball. Figure 3.1(*a*) shows the CAD drawing of the robot with its principal components marked. The body consists of three aluminium channels held together by reconfigurable circular decks and is about 1.5 m tall, with a diameter of 368 mm and a weight of about 52 kg. The ball consists of a hollow aluminium sphere of 185 mm diameter coated with polyurethane of 12.7 mm thickness. The robot is self-contained with all required components for operation on-board. The 48 V lead acid batteries are on the top deck, and can power the robot for a few hours. The robot's body also houses a battery charger on one of its top decks. A Crossbow VG700CA inertial measurement unit (IMU) with three fiber optic gyroscopes and three micro-electromechanical systems (MEMS) accelerometers is fixed on one of the lower decks on top of the ball drive unit. It provides Kalman-filtered roll and pitch angles of the body w.r.t. gravity, and also provides roll, pitch and yaw rates.

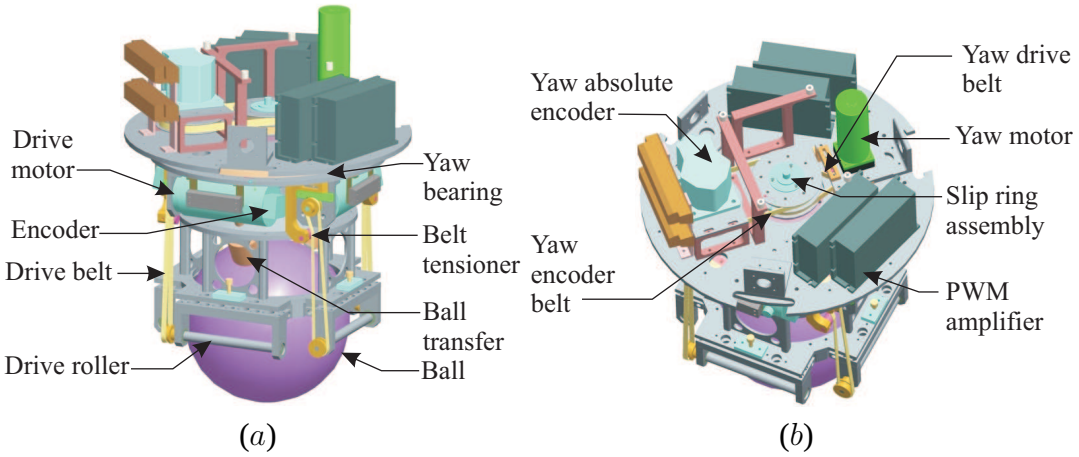


Figure 3.2: Four-motor inverse mouse-ball drive and yaw drive: (a) view showing main drive arrangement, (b) view showing yaw drive mechanism. (*Courtesy:* Ralph Hollis; Appeared in [76, 80])

3.2.1 Four-Motor Inverse Mouse-Ball Drive

The ball drive mechanism, shown in Fig. 3.2(a), is the inverse of an old-fashioned mouse-ball drive. It consists of four rollers, a pair each for orthogonal motion directions, actuated by four individual DC servomotors. The first version of the inverse mouse-ball drive [60] had a pair of drive and opposing passive rollers. This setup resulted in an undesirable "hopping motion" with the drive rollers producing an additional upward or downward force on the ball. This problem is avoided in the current design by actuating all four rollers of the inverse mouse-ball drive mechanism.

3.2.2 Yaw Mechanism

The ball drive mechanism is attached to the body using a large thin-section bearing, which allows yaw rotation of the body, $i.e.$, rotation about its vertical axis. The yaw drive mechanism, shown in Fig. 3.2(b), consists of a DC servomotor with planetary gears driving a pulley assembly at the center. An absolute encoder attached to the pulley assembly gives the orientation of the body frame w.r.t. the ball drive unit. A slip ring assembly at the center is used for drive motor currents and encoder signals, and allows unlimited yaw rotation of the body. The ballbot's first design in 2005 did not include the yaw drive mechanism, and it was added only in 2007.

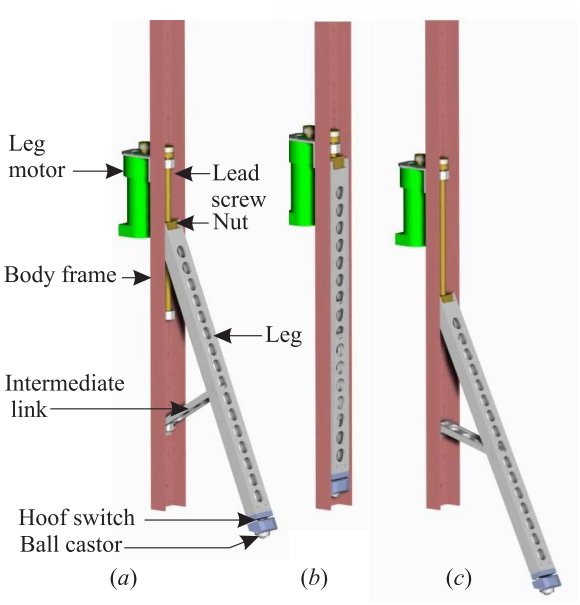


Figure 3.3: Leg drive: (a) Various components of the leg drive mechanism, (b) legs completed retracted, and (c) legs completely deployed. (*Courtesy:* Ralph Hollis; Appeared in [74, 80])

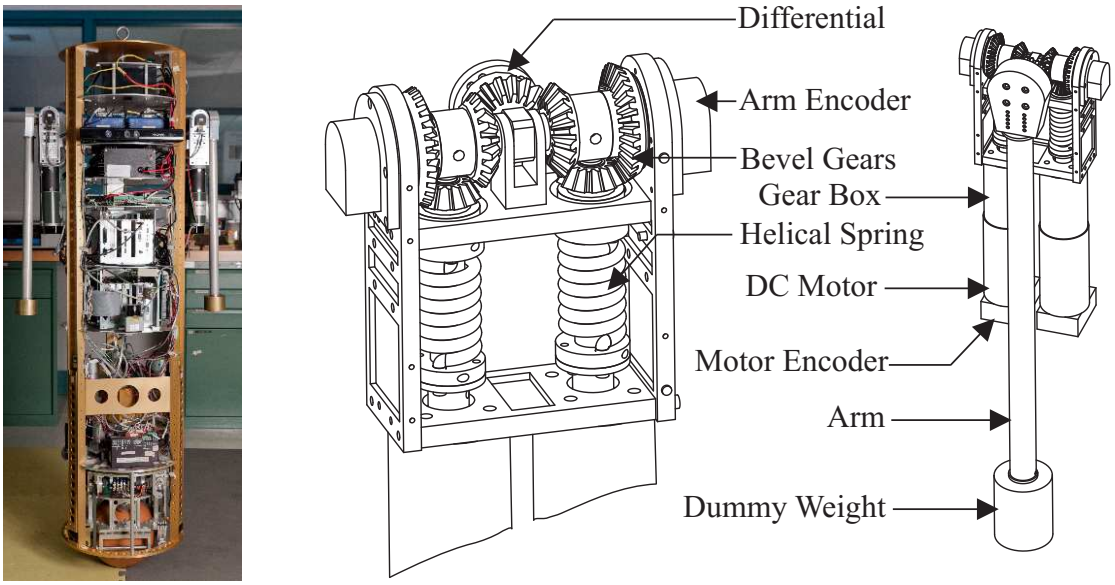
3.2.3 Legs

Each aluminium channel on the ballbot's body houses a leg that can be deployed to achieve static stability when powered down. The leg drive mechanism, shown in Fig. 3.3, consists of three independent DC servomotors, each driving a leg of length 0.48 m on a linear screw attached to the channel. The tip of each leg has a hoof switch, which signals its contact with the floor, and a ball caster, which allows the robot to roll on the floor when in a statically stable state. The legs enable the ballbot to switch between the statically stable state, with the legs down as shown in Fig. 3.1(c) to the dynamically stable state, balancing as shown in Fig. 3.1(b).

3.2.4 Arms

In 2011, Ralph Hollis and Byungjun Kim [81] designed and built a pair of 2-DOF arms for the ballbot as shown in Fig. 3.4(a). Each arm is an aluminium tube that is 0.457 m long and 0.89 mm thick with a changeable dummy weight (up to 2 kg) at its end. The arm attaches to its drive unit through a shoulder structure shown in Fig. 3.4(b).

Each arm is actuated by a pair of series-elastic actuators, each of which consists of a custom designed helical spring with a torsion coefficient of 16.37 Nm/rad, a brush DC motor with a



(a) (b)

Figure 3.4: (a) The ballbot with a pair of 2-DOF arms (*Courtesy:* Michael Shomin; Appeared in [73]), and (b) 2-DOF arm with series-elastic actuators (*Courtesy:* Ralph Hollis; Appeared in [73, 81]).

torque of 0.12 Nm at 3000 RPM, a 91:1 planetary gear train, and a 2000 counts per revolution (CPR) encoder. Each actuator connects with a pair of bevel gears of gear ratio 1:2, and a 1024 CPR optical encoder is attached to the end of the bevel gear shaft. The differential with three miter gears combines the torque from each actuator. The entire drive unit is fixed to one of the top decks just below the batteries. The trajectory tracking controllers for the arms were designed and tested by Byungjun Kim [81].

3.3 Dynamic Models

This section presents the equations of motion of the ballbot, one without arms and another with arms.

3.3.1 3D Ballbot without Arms

The ballbot without arms shown in Fig. 3.1 is modeled in 3D as a rigid cylindrical body on top of a rigid spherical wheel/ball with the following assumptions: (i) there is no slip between the

ball and the floor; and (ii) there is no yaw/spinning motion for both the body and the ball, $i.e.$, they have two degrees-of-freedom each. For ease of understanding the ballbot's configurations, a planar version of the ballbot with its planar configurations is shown in Fig. 3.5.

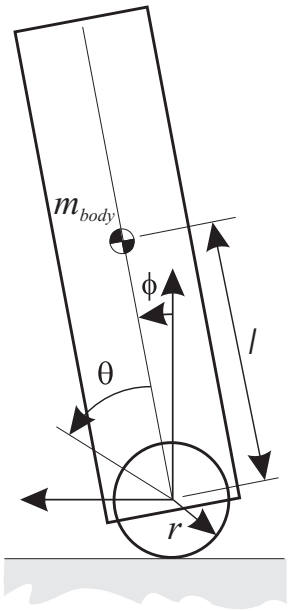


Figure 3.5: Planar ballbot model with ball and body configurations shown (Appeared in [74, 76, 78, 79, 80]).

The origin of the coordinate frame used to derive the dynamic model is at the center of the ball. The body angles (ϕ_x, ϕ_y) represent the roll and pitch angles of the body respectively w.r.t. the ball center. The ball angles (θ_x, θ_y) are chosen such that the x and y coordinates of the position of the ball center are given by $x = r(\theta_x + \phi_y)$ and $y = r(\theta_y - \phi_x)$. These ball angles correspond to the rotation measured by the encoders on the ball motors. The configuration vector $q \in \mathbb{R}^{4\times 1}$ is given by $q = [\theta_x, \theta_y, \phi_x, \phi_y]^T$.

The forced Euler-Lagrange equations of motion along with the friction terms are given by:

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}} - \frac{\partial \mathscr{L}}{\partial q} + D(\dot{q}) = \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}, \tag{3.1}$$

where, $\mathscr{L}(q, \dot{q}) = K(q, \dot{q}) - V(q)$ is the Lagrangian with kinetic energy $K(q, \dot{q})$ and potential energy $V(q)$, $D(\dot{q}) \in \mathbb{R}^{4\times 1}$ is the vector of frictional terms, and $\tau \in \mathbb{R}^{2\times 1}$ is the vector of gener-

alized forces. The forced Euler-Lagrange equations can be written in matrix form as follows:

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

where, $M(q) \in \mathbb{R}^{4\times4}$ is the mass/inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{4\times4}$ is the Coriolis and centrifugal matrix, and $G(q) \in \mathbb{R}^{4\times1}$ is the vector of gravitational forces. These system matrices are given below.

$$M(q) = \begin{bmatrix} \gamma_1 & 0 & -\gamma_2 S_{\phi_x} S_{\phi_y} & \gamma_1 + \gamma_2 C_{\phi_x} C_{\phi_y} \\ 0 & \gamma_1 & -\gamma_1 - \gamma_2 C_{\phi_x} C_{\phi_y} & 0 \\ -\gamma_2 S_{\phi_x} S_{\phi_y} & -\gamma_1 - \gamma_2 C_{\phi_x} & I_{xx} + \gamma_1 + \frac{\gamma_2 l}{r} + 2\gamma_2 C_{\phi_x} & -\gamma_2 S_{\phi_x} S_{\phi_y} \\ & & & I_{zz} + \gamma_1 + \gamma_3 C_{\phi_x}^2 \\ \gamma_1 + \gamma_2 C_{\phi_x} C_{\phi_y} & 0 & -\gamma_2 S_{\phi_x} S_{\phi_y} & +2\gamma_2 C_{\phi_x} C_{\phi_y} \end{bmatrix}, \tag{3.3}$$

$$C(q,\dot{q}) = \begin{bmatrix} 0 & 0 & -\gamma_2\left(C_{\phi_x} S_{\phi_y}\dot{\phi}_x + S_{\phi_x} C_{\phi_y}\dot{\phi}_y\right) & -\gamma_2\left(S_{\phi_x} C_{\phi_y}\dot{\phi}_x + C_{\phi_x} S_{\phi_y}\dot{\phi}_y\right) \\ 0 & 0 & \gamma_2 S_{\phi_x}\dot{\phi}_x & 0 \\ 0 & 0 & -\gamma_2 S_{\phi_x}\dot{\phi}_x & \gamma_3 S_{\phi_x} C_{\phi_x}\dot{\phi}_y \\ & & -\gamma_2 C_{\phi_x} S_{\phi_y}\dot{\phi}_x & -\left(\gamma_2 S_{\phi_x} C_{\phi_y} + \gamma_3 S_{\phi_x} C_{\phi_x}\right)\dot{\phi}_x \\ 0 & 0 & -\left(\gamma_2 S_{\phi_x} C_{\phi_y} + \gamma_3 S_{\phi_x} C_{\phi_x}\right)\dot{\phi}_y & -\gamma_2 C_{\phi_x} S_{\phi_y}\dot{\phi}_y \end{bmatrix}, \tag{3.4}$$

$$G(q) = \begin{bmatrix} 0 \\ 0 \\ -\frac{\gamma_2 g}{r} S_{\phi_x} C_{\phi_y} \\ -\frac{\gamma_2 g}{r} C_{\phi_x} S_{\phi_y} \end{bmatrix}, \tag{3.5}$$

where, $S_i = \sin(i)$, $C_i = \cos(i)$, $\gamma_1 = I_w + (m_b + m_w)r^2$, $\gamma_2 = m_b \ell r$ and $\gamma_3 = m_b \ell^2 + I_{yy}^b - I_{zz}^b$. The other symbols represent system parameters whose names and numerical values are shown in Table 3.1.

The vector of frictional terms $D(\dot{q})$ is given by:

$$D(\dot{q}) = \begin{bmatrix} D_c \text{sign}(\dot{\theta}_x) + D_v \dot{\theta}_x \\ D_c \text{sign}(\dot{\theta}_y) + D_v \dot{\theta}_y \\ 0 \\ 0 \end{bmatrix}, \tag{3.6}$$

where D_c is the Coulomb friction torque, and D_v is the viscous damping friction coefficient, whose numerical values are presented in Table 3.1. The vector of generalized forces $\tau \in \mathbb{R}^{2\times 1}$ is given by:

$$\begin{aligned} \tau &= J^T \begin{bmatrix} \tau_x^m \\ \tau_y^m \end{bmatrix} \\ &= \begin{bmatrix} \cos(\phi_y) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_x^m \\ \tau_y^m \end{bmatrix}, \end{aligned} \tag{3.7}$$

where, $J \in \mathbb{R}^{2\times 2}$ is the Jacobian matrix, τ_x^m and τ_y^m are the motor torques on the ball.

It is to be noted that the ball angles (θ_x, θ_y) form the actuated variables, whereas the body angles (ϕ_x, ϕ_y) form the unactuated variables. The last two equations of motion in Eq. 3.2, which correspond to the unactuated degrees of freedom are called *dynamic constraints*. These second-order differential equations form non-integrable constraints [88] on the dynamics of the system, and are also known as *second-order nonholonomic constraints*. The dynamic constraints for the 3D ballbot model without arms are given by the following two equations:

$$\begin{aligned} -\gamma_2 S_{\phi_x} S_{\phi_y} \ddot{\theta}_x - (\gamma_1 + \gamma_2 C_{\phi_x}) \ddot{\theta}_y + (I_{xx} + \gamma_1 + \frac{\gamma_2 l}{r} + 2\gamma_2 C_{\phi_x}) \ddot{\phi}_x - \gamma_2 S_{\phi_x} S_{\phi_y} \ddot{\phi}_y & \\ -\gamma_2 S_{\phi_x} \dot{\phi}_x^2 + \gamma_3 S_{\phi_x} C_{\phi_x} \dot{\phi}_y^2 - \frac{\gamma_2 g}{r} S_{\phi_x} C_{\phi_y} &= 0, \end{aligned} \tag{3.8}$$

$$\begin{aligned} (\gamma_1 + \gamma_2 C_{\phi_x} C_{\phi_y}) \ddot{\theta}_x - \gamma_2 S_{\phi_x} S_{\phi_y} \ddot{\phi}_x + (I_{zz} + \gamma_1 + \gamma_3 C_{\phi_x}^2 + 2\gamma_2 C_{\phi_x} C_{\phi_y}) \ddot{\phi}_y & \\ -\gamma_2 C_{\phi_x} S_{\phi_y} \dot{\phi}_x^2 - \gamma_2 C_{\phi_x} S_{\phi_y} \dot{\phi}_y^2 - 2(\gamma_2 S_{\phi_x} C_{\phi_y} + \gamma_3 S_{\phi_x} C_{\phi_x}) \dot{\phi}_x \dot{\phi}_y - \frac{\gamma_2 g}{r} C_{\phi_x} S_{\phi_y} &= 0, \end{aligned} \tag{3.9}$$

where, $S_i = \sin(i)$, $C_i = \cos(i)$, $\gamma_1 = I_w + (m_b + m_w) r^2$, $\gamma_2 = m_b \ell r$ and $\gamma_3 = m_b \ell^2 + I_{yy}^b - I_{zz}^b$. It can be seen from Eq. 3.8 and Eq. 3.9 that the dynamic constraint equations are independent of the position and velocity of the ball, *i.e.*, $(\theta_x, \theta_y, \dot{\theta}_x, \dot{\theta}_y)$.

3.3.2 3D Ballbot with Arms

The ballbot with arms is modeled as a rigid cylinder on top of a rigid sphere with a pair of massless arms having weights at their ends. The model makes the following assumptions: (i) there is no slip between the ball and the floor; and (ii) there is no yaw/spinning motion for either the ball or the body or the arms, $i.e.$, they have two degrees of freedom each. A planar version of the model along with its planar configurations is shown in Fig. 3.6.

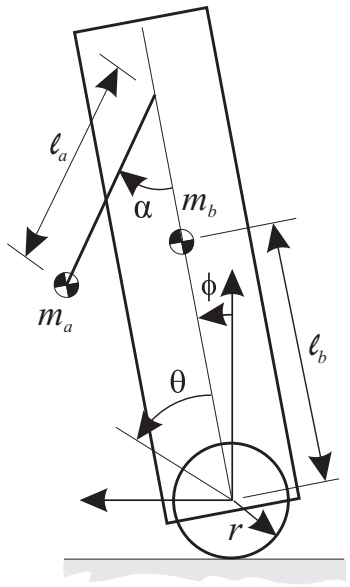


Figure 3.6: Planar configurations shown in a planar model of ballbot with arm (Appeared in [73, 81]).

There are eight configuration variables for the 3D ballbot model with arms represented by $q = [\theta, \alpha^l, \alpha^r, \phi]$, where, $\theta = [\theta_x, \theta_y]^T$ are configurations of the ball, $\alpha^l = [\alpha^l_x, \alpha^l_y]^T$ are configurations of the left arm, $\alpha^r = [\alpha^r_x, \alpha^r_y]^T$ are configurations of the right arm, and $\phi = [\phi_x, \phi_y]^T$ are configurations of the body. The forced Euler-Lagrange equations of motion of the ballbot with arms can be written in matrix form as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}, \tag{3.10}$$

where, $M(q) \in \mathbb{R}^{8\times 8}$ is the mass/inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{8\times 8}$ is the matrix of Coriolis and centrifugal terms, $G(q) \in \mathbb{R}^{8\times 1}$ is the vector of gravitational forces, and $\tau = [\tau_\theta, \tau_{\alpha_l}, \tau_{\alpha_r}]^T \in \mathbb{R}^{6\times 1}$ is the vector of generalized forces. The vector of body and arm configurations is represented

as $q_s = [\alpha^l, \alpha^r, \phi] \in \mathbb{R}^{6\times1}$. These variables are called *shape variables*, and their significance will be explained in Chapter 4. The system matrices in Eq. 3.10 are of the form given below:

$$M(q) = \begin{bmatrix} M_{\theta\theta} & M_{\theta\alpha^l}(q_s) & M_{\theta\alpha^r}(q_s) & M_{\theta\phi}(q_s) \\ M_{\alpha^l\theta}(q_s) & M_{\alpha^l\alpha^l}(q_s) & M_{\alpha^l\alpha^r}(q_s) & M_{\alpha^l\phi}(q_s) \\ M_{\alpha^r\theta}(q_s) & M_{\alpha^r\alpha^l}(q_s) & M_{\alpha^r\alpha^r}(q_s) & M_{\alpha^r\phi}(q_s) \\ M_{\phi\theta}(q_s) & M_{\phi\alpha^l}(q_s) & M_{\phi\alpha^r}(q_s) & M_{\phi\phi}(q_s) \end{bmatrix}, \tag{3.11}$$

$$C(q,\dot{q}) = \begin{bmatrix} \mathbf{0} & C_{\theta\alpha^l}(q_s,\dot{q}_s) & C_{\theta\alpha^r}(q_s,\dot{q}_s) & C_{\theta\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & C_{\alpha^l\alpha^l}(q_s,\dot{q}_s) & \mathbf{0} & C_{\alpha^l\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & \mathbf{0} & C_{\alpha^r\alpha^r}(q_s,\dot{q}_s) & C_{\alpha^r\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & C_{\phi\alpha^l}(q_s,\dot{q}_s) & C_{\phi\alpha^r}(q_s,\dot{q}_s) & C_{\phi\phi}(q_s,\dot{q}_s) \end{bmatrix}, \tag{3.12}$$

$$G(q) = \begin{bmatrix} \mathbf{0} \\ G_{\alpha^l}(q_s) \\ G_{\alpha^r}(q_s) \\ G_{\phi}(q_s) \end{bmatrix}, \tag{3.13}$$

where, each $M_{ij} \in \mathbb{R}^{2\times2}$, each $C_{ij} \in \mathbb{R}^{2\times2}$ and each $G_i \in \mathbb{R}^{2\times1}$. Equations 3.11 to 3.13 show that the system matrices are independent of the position and velocity of the ball, *i.e.*, $(\theta, \dot{\theta})$, and are dependent only on the shape variables and their velocities, *i.e.*, $\left(q_s, \dot{q}_s\right)$. The elements of all the submatrices shown above are presented in detail in Appendix A. It is to be noted that the body configurations ϕ are unactuated, whereas the rest of the configurations are actuated. The last two equations of motion corresponding to the unactuated variables ϕ form the *dynamic constraint* equations. These are *second-order nonholonomic constraints* as they are not even partially integrable [88]. The dynamic constraint equations in Eq. 3.10 can be written using the submatrices as follows:

$$\begin{aligned} M_{\phi\theta}(q_s)\ddot{\theta} + M_{\phi\alpha_l}(q_s)\ddot{\alpha}_l + M_{\phi\alpha_r}(q_s)\ddot{\alpha}_r + M_{\phi\phi}(q_s)\ddot{\phi} + C_{\phi\alpha_l}(q_s,\dot{q}_s)\dot{\alpha}_l \\ + C_{\phi\alpha_r}(q_s,\dot{q}_s)\dot{\alpha}_r + C_{\phi\phi}(q_s,\dot{q}_s)\dot{\phi} + G_{\phi}(q_s) \;=\; \mathbf{0}_{2\times1}. \end{aligned} \tag{3.14}$$

It can be seen from Eq. 3.14 that the dynamic constraint equations are independent of the position and velocity of the ball, *i.e.*, $\left(\theta, \dot{\theta}\right)$.

3.4 Parameter Estimation Experiments

This section presents a variety of experiments that were conducted on the ballbot to estimate its principal system parameters such that the dynamics of the 3D ballbot model better match the dynamics of the real robot. The design of these experimental setups, and the successful estimation of the system parameters are contributions of the work presented in this thesis.

3.4.1 Inertia Measurement

A torsional pendulum setup [132] was used to experimentally estimate moments of inertia of the body. The ballbot's body was suspended about its center of mass using a torsional spring as shown in Fig. 3.7, and its oscillations after an initial disturbance were observed. The angular velocity trajectory of the body obtained from the IMU, shown in Fig. 3.8, was used to find its frequency of oscillations.

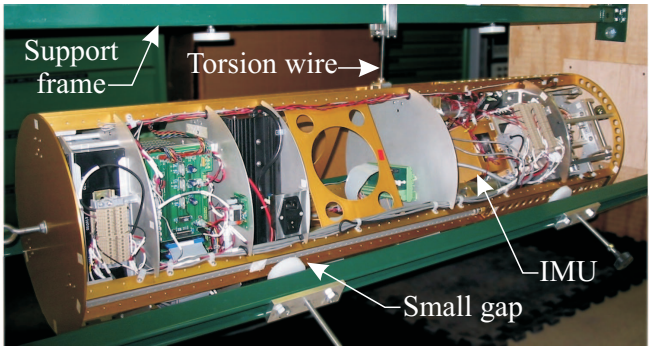


Figure 3.7: Torsional pendulum setup with ballbot suspended perpendicular to its length (*Courtesy:* Ralph Hollis; Appeared in [76, 80]).

The torsional spring constant was obtained by performing the same experiment with an I-beam whose moment of inertia was known. The torsional spring constant K is given by

$$K = I\omega_n^2, \tag{3.15}$$

where I is the moment of inertia of the suspended object, and ω_n is its natural frequency of oscillations. Therefore, the moment of inertia of the body about its center of mass is given by

$$I_{body} = I_{I-beam}\frac{\omega_{I-beam}^2}{\omega_{body}^2}. \tag{3.16}$$

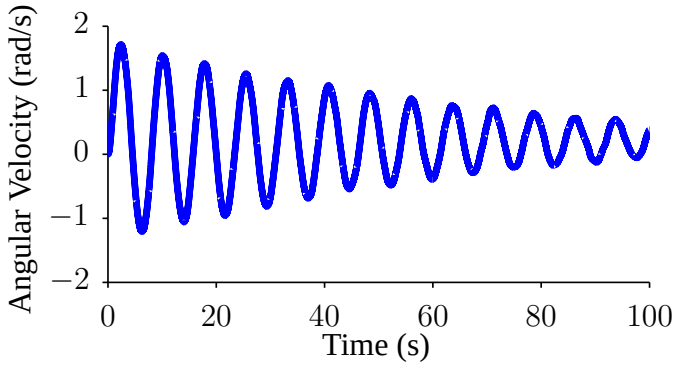


Figure 3.8: Damped sinusoidal oscillation used to determine the ballbot's moments of inertia (Appeared in [76, 80]).

The moments of inertia I^b_{xx} and I^b_{yy} of the ballbot's body were obtained by suspending it perpendicular to its length as shown in Fig. 3.7, whereas I^b_{zz} was obtained by hanging the ballbot's body vertically about its vertical axis. The estimated moment of inertia values from these experiments are shown in Table 3.1.

3.4.2 Friction Modeling

The Coulomb friction torque and the viscous friction coefficient were experimentally determined using the setup shown in Fig. 3.9, where the ballbot stood on a roller ball with its body constrained vertically. A ramp current input of slope m was given to the ball drive motors, and the angular velocity trajectory of the ball was recorded. The minimum current required to start the ball rolling is called the breakaway current, which when multiplied by the torque constant K_i of the drive unit gives the Coulomb friction torque D_c. The experiment was repeated with the current vector at 5° intervals.

After breakaway, the equation of motion of the ball can be written as

$$I_{ball}\ddot{\theta} = \tau(t) - \tau_v - D_c, \qquad \dot{\theta} > 0 \tag{3.17}$$

$$= K_i m t - D_v\dot{\theta} - D_c, \tag{3.18}$$

where τ_v is the viscous friction torque. The plot of the ball's angular velocity trajectory $\dot{\theta}(t)$ after breakaway can be approximated by a line of constant slope c [47] as shown in Fig. 3.10. Hence,

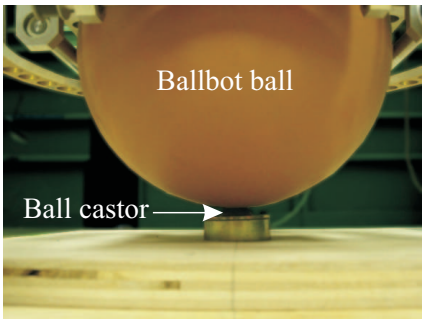


Figure 3.9: Ball rolling on the roller during friction tests (*Courtesy:* Ralph Hollis; Appeared in [76, 80]).

the angular velocity can be written as

$$\dot{\theta} = ct - d, \quad \dot{\theta} > 0 \tag{3.19}$$

$$\ddot{\theta} = c. \tag{3.20}$$

Solving Eq. 3.18−3.20, we get

$$D_v = \frac{K_i m}{c}. \tag{3.21}$$

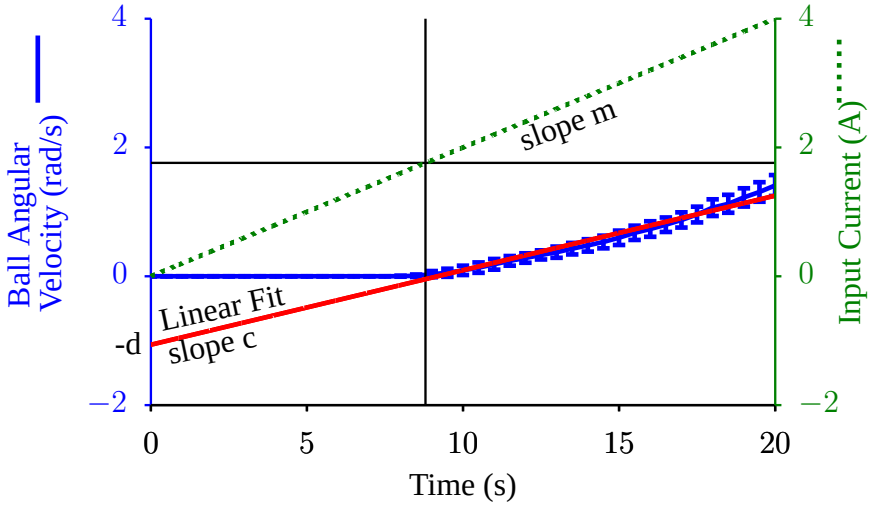
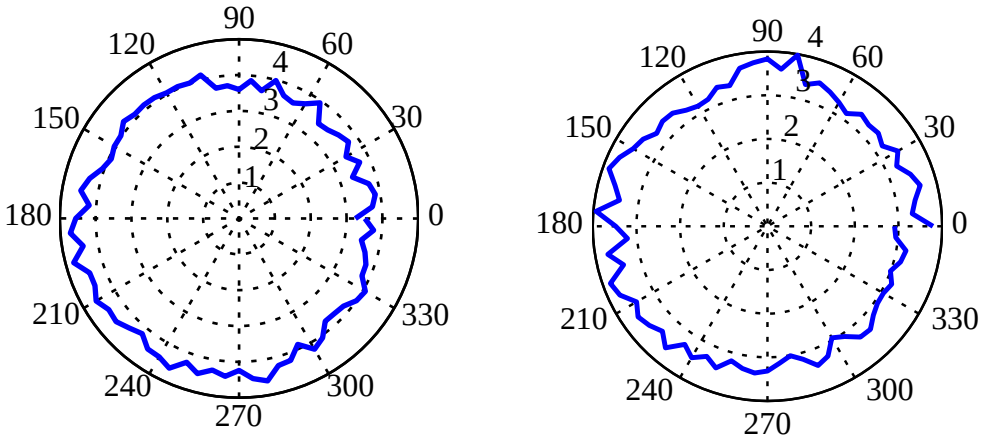


Figure 3.10: Ball response to the ramp current inputs to the ball drive motors used for determing coulomb and viscous friction terms (Appeared in [80]).

The radial plots of Coulomb friction torque and viscous friction coefficient in different drive

directions are shown in Fig. 3.11. Table 3.1 presents the average coulomb friction torque and viscous friction coefficient values that are used in simulation.



(a) (b)

Figure 3.11: Radial plots as functions of drive directions: (a) Coulomb friction torque D_c (Nm); (b) Viscous friction coefficient D_v (Nms/rad). (Appeared in [80])

3.5 Control Architecture

This section provides a brief description of the different controllers used on the ballbot. The design and successful implementation of these controllers are contributions of the work presented in this thesis. The videos of the ballbot achieving the experimental results presented in this section can be found in Video D.1.

3.5.1 Balancing Control

The balancing controller, for obvious reasons, is the single most important controller on the robot. The balancing controller takes desired body angles, $i.e.$, the roll and pitch angles of the body, as inputs, and balances the robot about these desired body angles. The desired body angles are zero for a pure balancing operation, $i.e.$, standing still. Since the body angles are unactuated, the balancing controller cannot directly track the desired body angles. The balancing controller indirectly achieves this objective by actuating the ball such that the projection of the body's

Table 3.1: System Parameters

Parameter	Symbol	Value
Z-axis CM from ball center	ℓ	0.69 m
Ball radius	r	0.106 m
Ball mass	m_w	2.44 kg
Ball inertia	I_w	0.0174 kgm^2
Roll moment of inertia about CM	I^b_{xx}	12.59 kgm^2
Pitch moment of inertia about CM	I^b_{yy}	12.48 kgm^2
Yaw moment of inertia about CM	I^b_{zz}	0.66 kgm^2
Body mass	m_b	51.66 kg
Coulomb friction torque	D_c	3.82 Nm
Viscous damping friction coefficient	D_v	3.68 Nms/rad
Torque constant for the ball drive	K_i	2.128 Nm/A

center of mass on the floor tracks the projection of the desired center of mass obtained from desired body angles as shown in Fig. 3.12.

The balancing controller consists of two independent controllers, one for each of the vertical planes. Each one is a Proportional-Integral-Derivative (PID) controller whose gains were tuned manually. When the real ballbot balances, the variation of its pitch angle remains within $\pm 0.05^\circ$ as shown in Fig. 3.13. Similar results were obtained for the roll angle.

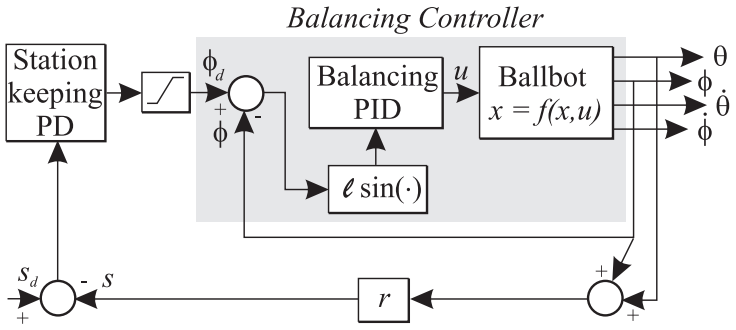


Figure 3.12: Block diagram for the station keeping controller with the balancing control block (Appeared in [76, 80]).

3.5.2 Outer Loop Control

The balancing controller is good at balancing about desired body angles but does not achieve any desired ball position on the floor. This is achieved by using an outer control loop around the bal-

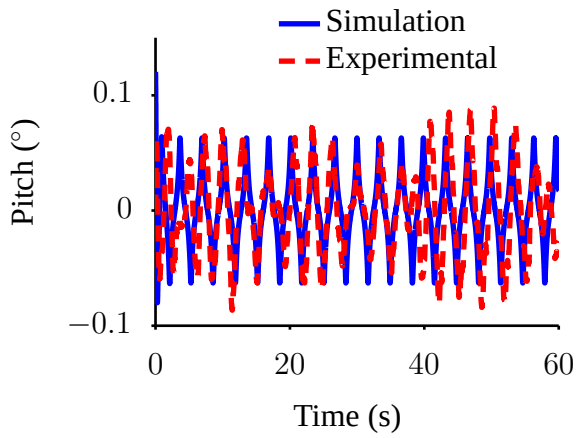
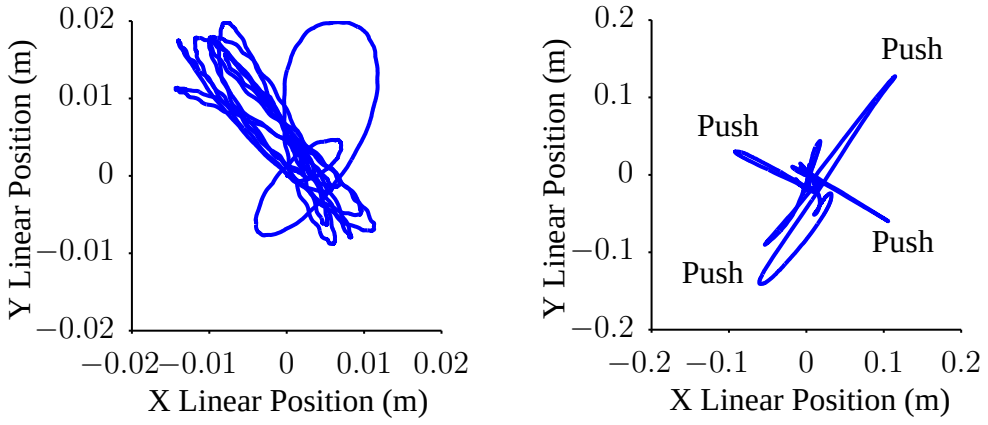


Figure 3.13: Pitch angle of the ballbot's body while balancing about a zero desired body angle (Appeared in [76, 80]).

ancing controller, as shown in Fig. 3.12. The outer loop controller provides desired body angles to the balancing controller. This section presents two outer-loop controllers: (i) stationkeeping control, and (ii) velocity control.

Stationkeeping is the act of balancing at a desired position even when disturbed. The stationkeeping controller is a Proportional-Derivative (PD) controller that outputs desired body angles depending on the error between the ball's current and desired positions. The PD controller's angle outputs are saturated to avoid large lean angles, and its gains were tuned manually. Figure 3.14(a) shows an XY plot of the position of the ball on a carpeted floor. As one can see, the balancing controller is able to keep the ball close to its starting point on the floor to within about $\pm$10-15 mm when the robot is not disturbed. Unlike the balancing controller, the stationkeeping controller keeps the ball close to its starting point even when disturbed. The XY plot of the ball's position when the body is pushed in all four directions is shown in Fig. 3.14(b).

The velocity controller is a manually tuned Proportional-Integral (PI) controller that outputs desired body angles depending on the error between the ball's current and desired velocities. The velocity controller is concerned only with the ball's velocity and does not bother about its position. The velocity controller has two major applications, one as a stopping controller, which enables the ballbot to slow down and come to rest when subjected to large disturbances; and the other for teleoperation of the ballbot wherein the user can provide velocity commands using a joystick. Just like the stationkeeping controller, the angle outputs from the velocity controller are also saturated to avoid large lean angles.



(*a*) Balancing Control (*b*) Stationkeeping Control

Figure 3.14: Balancing at a position: (a) ball track on the carpeted floor using only the balancing controller, (b) operation of the station keeping controller when the body is pushed off its position. (Appeared in [76, 80])

3.5.3 Yaw Control

In the present control architecture, the yaw controller is decoupled from the balancing controller for simplicity and ease of control. It consists of two loops: an inner PI control loop that feeds back the yaw angular velocity $\dot{\psi}$, and an outer PD control loop that feeds back both the yaw angle ψ and the yaw angular velocity $\dot{\psi}$, as shown in Fig. 3.15. The desired angular velocity output from the outer-loop position controller is saturated to avoid high angular velocities that could potentially drive the balancing system unstable.

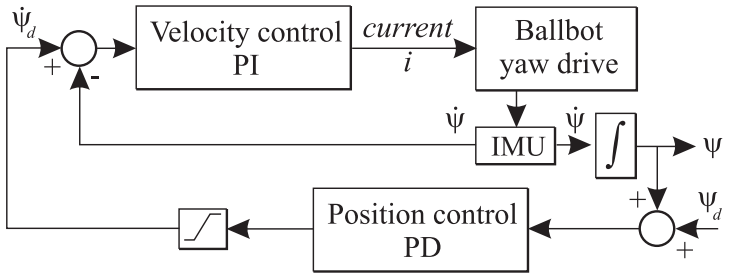


Figure 3.15: Block diagram of the yaw controller (Appeared in [76, 80]).

During the yaw motion, the IMU attached to the body frame rotates, while the ball drive unit does not. The angle offset χ between the drive unit and the body frame is given by the absolute yaw encoder. The balancing controller requires a corresponding rotation transformation

to be performed on the roll and pitch angles from the IMU in order to transform them into the coordinate frame of the ball drive unit. Figure 3.16 shows selected frames of the ballbot performing a 360° yaw motion while balancing, and the corresponding yaw angular plot is shown in Fig. 3.17. The results show the capability of the robot to rotate in place, which will be useful when the ballbot uses its arms to manipulate objects. This thesis does not present any work on navigation of the ballbot using the yaw motion. For all the experimental results presented in Chapter 4 and Chapter 5, the yaw controller was used to ensure that the body did not yaw while the ballbot was in motion.

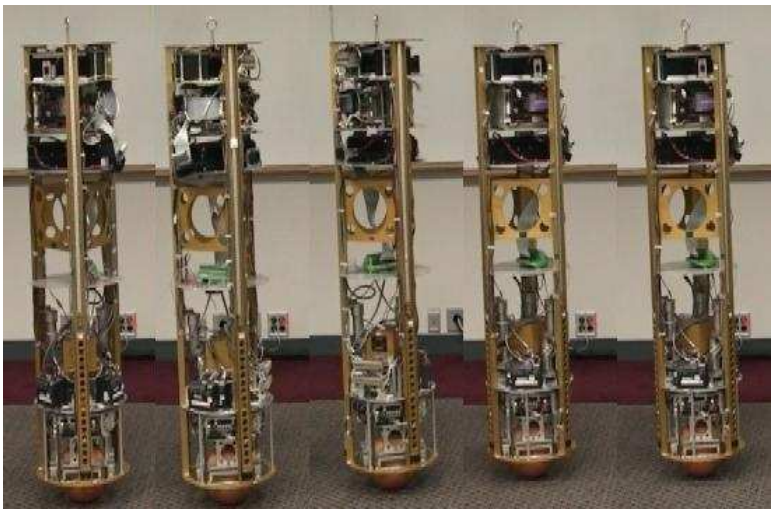


Figure 3.16: Selected frames of 360° yaw motion video (Appeared in [76, 80]).

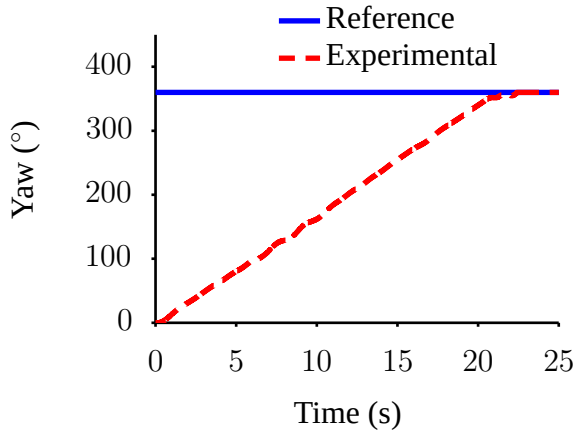


Figure 3.17: 360° yaw motion of the ballbot's body while balancing (Appeared in [76, 80]).

3.5.4 Leg Control

As mentioned in Sec. 3.2, the ballbot deploys its three legs to achieve static stability. This implies that the ballbot has two states in terms of stability: (i) a Dynamically Stable State (DSS), in which it balances on the ball; and (ii) a Statically Stable State (SSS), in which it rests with all three legs fully deployed. The ballbot is capable of moving in both these states [66]. In DSS, the ballbot moves around by balancing on the ball and leaning its body in the desired direction of motion, whereas in SSS, the robot moves by rolling the ball with all its three legs sliding on the floor. The ball caster at the tip of each leg makes this sliding possible. However, in SSS, the motion is restricted to only smooth planar surfaces. The leg control plays a vital role in the automatic transition between DSS and SSS, which is a requirement for a fully autonomous ballbot.

All three legs have independent controllers for both lifting and deploying operations. The legs-up controller is a PI velocity controller that stops when the legs hit the body, *i.e.*, the leg motors stall. The legs-down controller uses two control loops similar to the yaw controller shown in Fig. 3.15. The inner PI control loop feeds back the velocity of the leg, and the outer PD control loop feeds back both the position and velocity of the leg. The legs-down controller stops when the hoof switches at the tip of the legs hit the floor.

The legs-up controller and the balancing controller can be scheduled to operate together such that the ballbot transitions from SSS to DSS. This was first demonstrated by Anish Mampetta [66]. However, the simultaneous operation of the legs-up controller and the balancing controller created large transients because the ballbot's body is not always vertical (zero body angles) when the legs are down. In order to achieve smooth transition, the body angles must be close to zero before lifting the legs up. This is achieved using the legs-adjust controller shown in Fig. 3.18, which is a contribution of the work presented in this thesis. When all three legs are fully down, the legs-adjust controller changes the position of the legs to achieve zero body angles. This is done with the knowledge that the three legs and the body form an overconstrained spatial linkage [129].

The top view of the ballbot with all three legs deployed is shown in Fig. 3.19(a). The spatial linkage consisting of the ballbot and the three legs attached to the floor with PR joints was simulated in Open Dynamics Engine. For each leg, the leg nut was moved up and the effect of the position of the leg nut on the body angles (both roll and pitch) was recorded. In Fig. 3.19(a), it can be seen that the position of leg 1 affects only the pitch (rotation about y axis) and not the roll, while the positions of legs 2 and 3 affect both. A graph showing the position of leg 1 as a function of the body pitch angle is shown in Fig. 3.19(b). Similar graphs hold for legs 2 and 3.

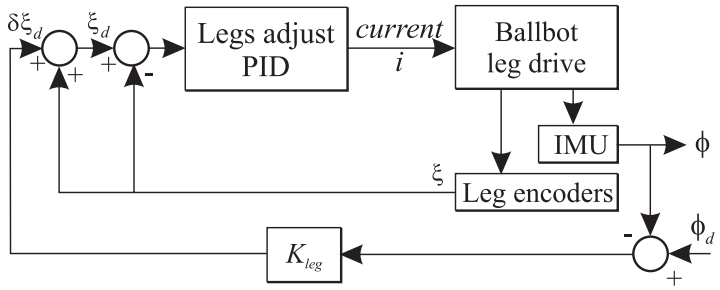


Figure 3.18: Block diagram for the legs-adjust controller (Appeared in [76, 80]).

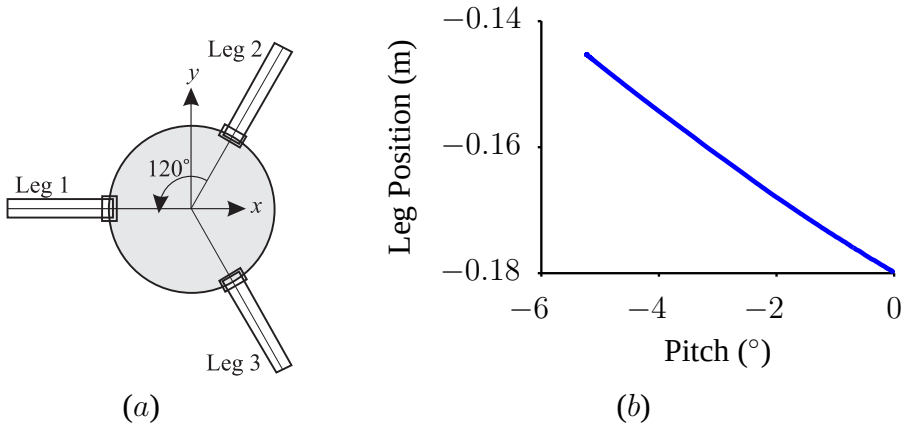


Figure 3.19: (a) Top view of the ballbot with all three legs deployed; (b) Position of leg 1 as a function of body pitch. (Appeared in [76, 80])

The legs-adjust controller controls the position of leg 1 to achieve the desired pitch, and controls the positions of legs 2 and 3 to achieve the desired roll. As can be seen in Fig. 3.19(b), the relationship between the leg position ξ and the body angle ϕ is approximately linear of the form $\xi = K_{leg}\phi + c_{leg}$. This relationship was used to create a PID controller that adjusts the positions of the legs so as to tilt the body to desired roll and pitch angles as shown in Fig. 3.18. This controller facilitates a smooth automatic transition from SSS to DSS as the initial body angles can be adjusted to be close to zero. While transitioning from DSS to SSS, the balancing controller is turned off when the hoof switches on the legs contact the floor. A flow chart of this automatic transition procedure is shown in Fig. 3.20. Figure 3.21 shows selected frames from a video of the ballbot automatically transitioning from SSS to DSS, and vice versa.

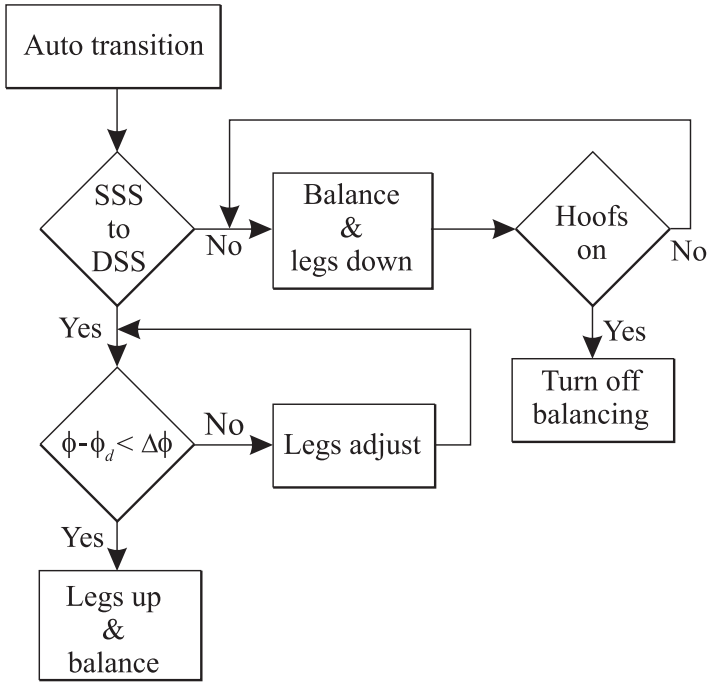


Figure 3.20: Flow chart for the automatic transition operation (Appeared in [76, 80]).

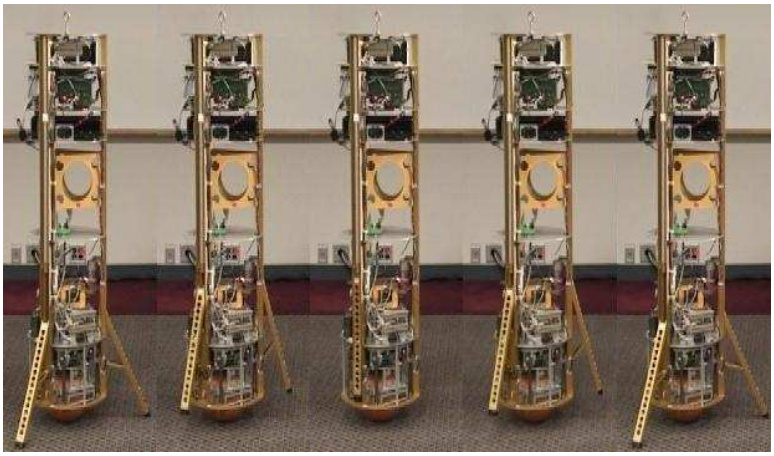


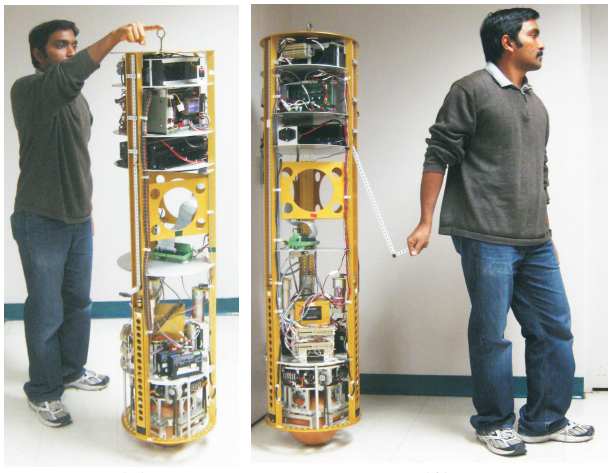
Figure 3.21: Selected frames of the automatic transition from SSS to DSS, and vice versa (Appeared in [76, 80]).

3.6 Human–Ballbot Physical Interaction

Humans are physically interactive with everything in their environments and hence, robots operating in human environments must handle physical interactions with humans and objects [106]. The dynamic stability of balancing mobile robots like the ballbot naturally enable them to achieve several interesting physically interactive behaviors [75], some of which are described below. The videos of the ballbot performing these physically interactive behaviors can be found in Video D.2.

3.6.1 Ease of Mobility

Humans should be able to physically move a robot operating in their environments with ease. The ballbot, while balancing, can be physically moved around with little effort. It is generally a difficult task to physically move a heavy statically stable mobile robot, whereas the ballbot can be moved around with just a single finger as shown in Fig. 3.22(a). In addition to this, humans should also be able to physically stop and control a robot operating in their environments with minimal force. The ballbot can be stopped with little effort even while it is in motion. It can also be dragged around using a passive lever hand that is attached to one of its channels as shown in Fig. 3.22(b).



(a) (b)

Figure 3.22: Moving the ballbot: (a) with a finger; (b) with a passive lever hand (Appeared in [75, 80]).

3.6.2 Robustness

Robots operating in cluttered and/or crowded human environments should be robust to disturbances, user mistakes, and even ill-treatment to a certain degree. The ballbot is robust to large disturbances like even shoves and kicks as shown in Fig. 3.23. The ballbot can also handle collisions with furniture and walls.

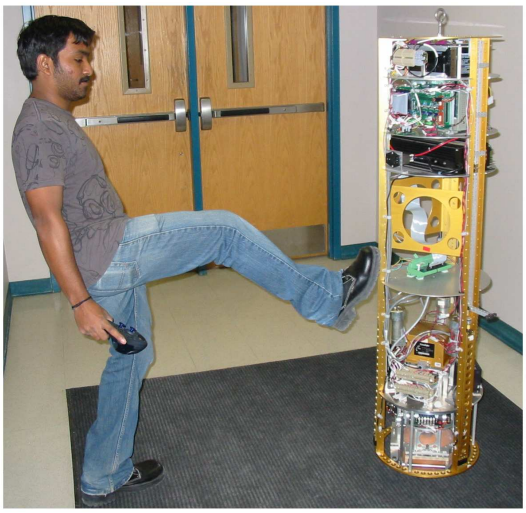


Figure 3.23: Kicking the ballbot (Appeared in [75, 80]).

3.6.3 Human Intent Detection

The force exerted by a human on the ballbot directly corresponds to its acceleration, which can be used to detect certain basic intentions. For example, a soft push can be considered as unintentional, whereas a hard push can be interpreted as a move away command. The ballbot's response to such human intentions is shown in Fig. 3.24. When given a soft push, the ballbot continues to stationkeep at its current position ①, whereas given a hard push, it moves away and stationkeeps at a different position ② on the floor.

3.6.4 Learn and Repeat

One can use physical interactions to teach different tasks to robots. This section presents some initial results of a learn and repeat behavior using the ballbot. In the learn mode, the human user physically moves the ballbot in a desired path, and in the repeat mode, the ballbot attempts to

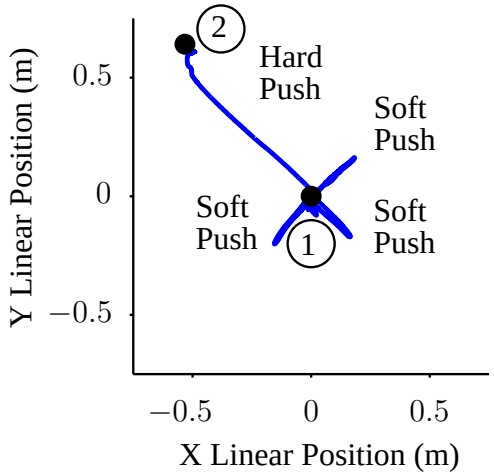


Figure 3.24: Human Intent Detection (Appeared in [75, 80]).

track the learned path. The ballbot's attempts at repeating approximate linear and circular paths learned from the human user are shown in Fig. 3.25.

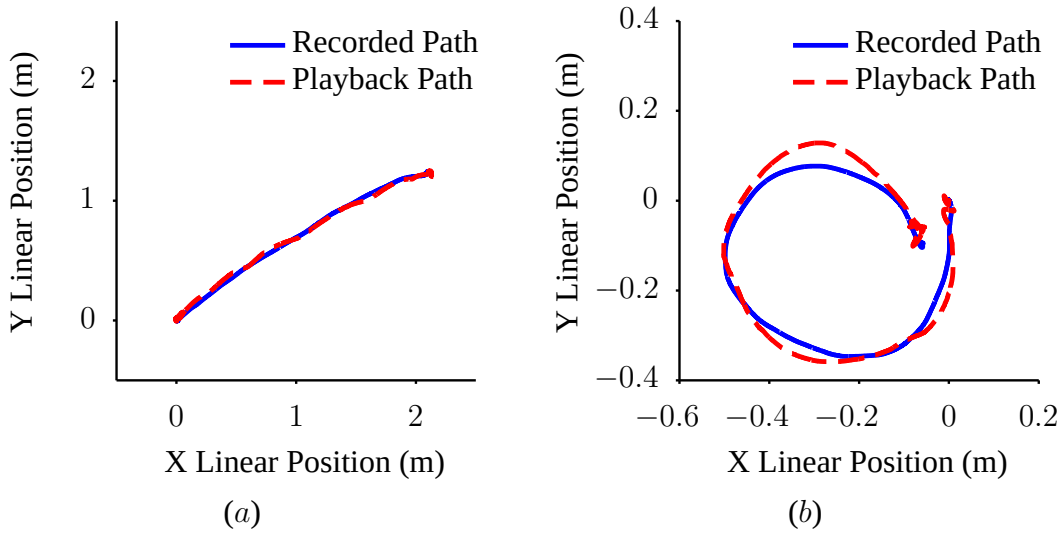


Figure 3.25: Learn-Repeat behavior: (Approximate) (a) Linear and (b) Circular Motion. (Appeared in [75, 80])

3.6.5 Ballbot Interface and Teleoperation

The ballbot has a highly interactive graphical user interface that allows wireless teleoperation of the robot using a joystick. A high-level overview of the ballbot's software architecture for the

entire work presented in this thesis is shown in Fig. C.1. In addition to the behaviors presented above [75], the ballbot has been reliably teleoperated at fast walking speeds for hundreds of meters over surfaces ranging from vinyl tile to carpet to rough concrete to metal gratings. The ballbot was also successfully teleoperated on ramps with angles up to about 4°. The ballbot was able to drive into and out of elevators and over the cracks and misalignment between elevator cars and floors with ease. The ballbot was also able to drive up and down a two-story helical ramp with ease.

3.7 Summary

This chapter introduced the ballbot developed by Ralph Hollis at Carnegie Mellon University, Pittsburgh, USA. It presented the history of the ballbot, and details of its hardware. It also presented the dynamic models of the 3D ballbot with and without arms, and the parameter estimation experiments that were conducted to estimate the principal system parameters. This chapter then described the control architecture for the ballbot, which used a balancing controller to stabilize the system, and an outer loop controller for stationkeeping and velocity control. It also described the controllers developed for yaw control and leg control. In addition to the legs-up and the legs-down controllers, a controller that adjusts the positions of the legs to enable smooth automatic transition from the statically stable state to the dynamically stable state was also presented. The balancing controller enabled the ballbot to be robust to disturbances including pushes, kicks and even collisions with walls and furniture. It also enabled the ballbot to be physically interactive. This chapter also presented experimental results demonstrating some interesting human-robot physical interaction behaviors with the ballbot. The contribution of the work presented in this thesis is not in the design of the hardware, but in the design of the controllers that enabled the ballbot to balance and operate reliably.

Chapter 4

Planning in Shape Space

Balancing mobile robots like the ballbot are capable of moving with speed and grace comparable to that of humans because of their dynamic stability. This chapter presents trajectory planning algorithms that exploit the natural dynamics of balancing mobile robots like the ballbot to achieve desired fast and dynamic motions (answer to RQ 1). This chapter introduces a special class of underactuated systems called *shape-accelerated balancing systems* to which balancing mobile robots like the ballbot belong. It then presents a trajectory planner that plans motions in the shape space, which when tracked will result in optimal tracking of desired fast and dynamic motions in the position space.

4.1 Underactuated Mechanical Systems

The forced Euler-Lagrange equations of motion for a mechanical system are given by:

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}} - \frac{\partial \mathscr{L}}{\partial q} = \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}, \tag{4.1}$$

where $q \in \mathbb{R}^n$ is the configuration vector, $\mathscr{L}(q, \dot{q}) = K(q, \dot{q}) - V(q)$ is the Lagrangian with kinetic energy $K(q, \dot{q})$ and potential energy $V(q)$, and $\tau \in \mathbb{R}^m$ is the vector of generalized forces.

A mechanical system satisfying Eq. 4.1 is said to be an *underactuated system* [118] if $m < n$, $i.e.$, there are fewer independent control inputs than configuration variables. Equation 4.1 can be

written in matrix form as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}, \tag{4.2}$$

where $M(q) \in \mathbb{R}^{n\times n}$ is the mass/inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal matrix, and $G(q) \in \mathbb{R}^n$ is the vector of gravitational forces.

The ballbot without arms, shown in Fig. 3.1(b), has four configuration variables given by $q = [\theta, \phi]^T \in \mathbb{R}^{4\times 1}$, where $\theta = [\theta_x, \theta_y]^T \in \mathbb{R}^{2\times 1}$ are the ball angles, and $\phi = [\phi_x, \phi_y]^T \in \mathbb{R}^{2\times 1}$ are the body angles. The ballbot with a pair of 2-DOF arms, shown in Fig. 3.4(a), has four additional configuration variables, namely, the left arm angles $\alpha^l = [\alpha^l_x, \alpha^l_y]^T \in \mathbb{R}^{2\times 1}$, and the right arm angles $\alpha^r = [\alpha^r_x, \alpha^r_y]^T \in \mathbb{R}^{2\times 1}$. The body configurations are unactuated, whereas the ball and arm configurations are actuated. Therefore, the ballbot without arms has two actuated and two unactuated configurations, $i.e.$, $n = 4$ and $m = 2$, while the ballbot with arms has six actuated configurations and two unactuated configurations, $i.e.$, $n = 8$ and $m = 6$.

4.1.1 Position and Shape Variables

The configuration variables $q \in \mathbb{R}^n$ of any dynamic system can be split into *position variables* $q_x \in \mathbb{R}^{n_x}$, and *shape variables* $q_s \in \mathbb{R}^{n_s}$, $i.e.$, $q = [q_x, q_s]^T$ and $n_x + n_s = n$. The shape variables q_s are those that appear in the mass/inertia matrix $M(q)$, whereas the position/external variables q_x are those that do not appear in the mass/inertia matrix $M(q)$. This implies that $M(q)$ is a function of only the shape variables q_s.

For the ballbot with arms, the ball angles form the position variables, $i.e.$, $q_x = \theta \in \mathbb{R}^{2\times 1}$, whereas the arm and body angles form the shape variables, $i.e.$, $q_s = [\alpha^l, \alpha^r, \phi]^T \in \mathbb{R}^{6\times 1}$. It can be seen that the ballbot with arms has more shape variables than position variables. For the ballbot without arms, only the body angles form the shape variables, $i.e.$, $q_s = \phi \in \mathbb{R}^{2\times 1}$ and hence, there are equal number of position and shape variables.

Since the mass/inertia matrix $M(q)$ is a function of only the shape variables q_s, it is independent of the position variables q_x. This implies that the kinetic energy $K(q,\dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q}$ is also independent of the position variables q_x. In this case, the Lagrangian system is said to have kinetic symmetry [85]. The system matrices in Eq. 4.2 can be split into submatrices based on the

position and shape variables as follows:

$$M(q_s) = \begin{bmatrix} M_{xx}(q_s) & M_{xs}(q_s) \\ M_{sx}(q_s) & M_{ss}(q_s) \end{bmatrix}, \tag{4.3}$$

$$C(q, \dot{q}) = \begin{bmatrix} C_{xx}(q, \dot{q}) & C_{xs}(q, \dot{q}) \\ C_{sx}(q, \dot{q}) & C_{ss}(q, \dot{q}) \end{bmatrix}, \tag{4.4}$$

$$G(q) = \begin{bmatrix} G_x(q) \\ G_s(q) \end{bmatrix}. \tag{4.5}$$

4.1.2 Shape-Accelerated Balancing Systems

The work presented in this chapter focuses on a special class of underactuated systems called *shape-accelerated balancing systems* [72, 73], for which the ballbot is an example. The introduction of shape-accelerated balancing systems and a trajectory planner that exploits the special properties of such systems to plan motions in the shape space in order to achieve desired motions in the position space are contributions of the work presented in this thesis. Shape-accelerated balancing systems are underactuated mechanical systems that satisfy the following properties:

(a) The number of unactuated variables equals the number of position variables $q_x \in \mathbb{R}^{n_x}$, $i.e.$, $n_x = n - m$. This implies that the number of dynamic constraint equations equals the number of position variables.

The work presented in this chapter primarily deals with systems whose position variables $q_x \in \mathbb{R}^{n_x}$ are actuated, while their shape variables $q_s \in \mathbb{R}^{n_s}$ contain both actuated $q_{s_a} \in \mathbb{R}^{n_{s_a}}$ and unactuated variables $q_{s_u} \in \mathbb{R}^{n_{s_u}}$, $i.e.$, $n_{s_a} + n_{s_u} = n_s$, $n_x + n_{s_a} = m$ and $n_{s_u} = n - m$. For example, in the case of the ballbot with arms, the ball angles form the actuated position variables, $i.e.$, $q_x = \theta \in \mathbb{R}^{2\times 1}$, the arm angles form the actuated shape variables, $i.e.$, $q_{s_a} = [\alpha^l, \alpha^r]^T \in \mathbb{R}^{4\times 1}$, and the body angles form the unactuated shape variables, $i.e.$, $q_{s_u} = \phi \in \mathbb{R}^{2\times 1}$. It can be seen that the number of unactuated variables (body angles) equals the number of position variables (ball angles).

However, underactuated mechanical systems with unactuated position variables and actuated shape variables like the marble-maze robot also form examples of shape-accelerated balancing systems, and the trajectory planner presented in this chapter can also be applied to such systems.

(b) The number of shape variables is an integral multiple of the number of position variables, *i.e.*, $n_s = k\ n_x$, for some $k \in \mathbb{Z}^+$, where $\mathbb{Z}^+$ represents the set of positive integers. This work defines a *shape set* to be a set of n_x shape variables that can independently affect the dynamics of all the position variables. Since the number of shape variables in each shape set equals the number of position variables, a system with actuated and unactuated shape variables has one unactuated shape set and $k-1$ actuated shape sets.
For example, the ballbot with arms has one unactuated shape set formed by its body angles $\phi \in \mathbb{R}^{2\times 1}$, and two actuated shape sets formed by its left arm angles $\alpha^l \in \mathbb{R}^{2\times 1}$ and right arm angles $\alpha^r \in \mathbb{R}^{2\times 1}$.

(c) The potential energy $V(q)$ is independent of the position variables q_x, which implies that the vector of gravitational forces $G(q) = \dfrac{\partial V(q)}{\partial q} \in \mathbb{R}^{n\times 1}$ is also independent of the position variables q_x.
Since both kinetic and potential energies are independent of the position variables q_x, the Lagrangian $\mathscr{L}$ is also independent of the position variables q_x, *i.e.*, $\mathscr{L}$ is symmetric w.r.t. the position variables q_x.

(d) The vector of Coriolis and centrifugal forces given by $C(q,\dot{q})\dot{q}$ is independent of both the position and velocity of the position variables, *i.e.*, q_x and $\dot{q}_x$. Since the mass/inertia matrix $M(q)$ is independent of the position variables q_x, the Coriolis and centrifugal matrix $C(q,\dot{q})$, whose elements are derived from the elements of $M(q)$, is also independent of the position variables q_x. The vector $C(q,\dot{q})\dot{q}$ can be independent of $\dot{q}_x$ only if $C_{xx}(q,\dot{q}) = \mathbf{0} \in \mathbb{R}^{n_x\times n_x}$, $C_{sx}(q,\dot{q}) = \mathbf{0} \in \mathbb{R}^{n_s\times n_x}$, and the matrices $C_{xs}(q,\dot{q}) \in \mathbb{R}^{n_x\times n_s}$ and $C_{ss}(q,\dot{q}) \in \mathbb{R}^{n_s\times n_s}$ are both independent of $\dot{q}_x$. These conditions are achieved when $M_{xx}(q_s) \in \mathbb{R}^{n_x\times n_x}$ is constant; and $M_{sx}(q_s) \in \mathbb{R}^{n_s\times n_x}$ has differentially symmetric rows [85], *i.e.*, $\dfrac{\partial M_{sx}(q_s)}{\partial q_s^i} = \left(\dfrac{\partial M^i{}_{sx}(q_s)}{\partial q_s}\right)^T$, where $M^i_{sx}(q_s)$ refers to the i^{th} row of $M_{sx}(q_s)$.
Since $M(q)$, $C(q,\dot{q})\dot{q}$ and $G(q)$ are all independent of both q_x and $\dot{q}_x$, the equations of motion shown in Eq. 4.2 are independent of both q_x and $\dot{q}_x$. For underactuated systems with actuated and unactuated shape variables satisfying properties (a)–(d), the matrices in Eq. 4.2 can be written as:

$$M(q_s) = \begin{bmatrix} M_{xx} & M_{xs_a}(q_s) & M_{xsu}(q_s) \\ M_{s_ax}(q_s) & M_{s_as_a}(q_s) & M_{s_as_u}(q_s) \\ M_{s_ux}(q_s) & M_{s_us_a}(q_s) & M_{s_us_u}(q_s) \end{bmatrix}, \tag{4.6}$$

$$C(q_s, \dot{q}_s) = \begin{bmatrix} \mathbf{0} & C_{xs_a}(q_s, \dot{q}_s) & C_{xs_u}(q_s, \dot{q}_s) \\ \mathbf{0} & C_{s_a s_a}(q_s, \dot{q}_s) & C_{s_a s_u}(q_s, \dot{q}_s) \\ \mathbf{0} & C_{s_u s_a}(q_s, \dot{q}_s) & C_{s_u s_u}(q_s, \dot{q}_s) \end{bmatrix}, \tag{4.7}$$

$$G(q) = \begin{bmatrix} \mathbf{0} \\ G_{s_a}(q_s) \\ G_{s_u}(q_s) \end{bmatrix}. \tag{4.8}$$

(e) The Jacobian linearization of the system is controllable [44] at the origin, where the origin is the unstable equilibrium. The system in Eq. 4.2 can be written in state-space form with affine control as $\dot{x} = f(x) + g(x)u$, where $x = [q, \dot{q}]^T \in \mathbb{R}^{2n}$ is the state vector and $u = \tau \in \mathbb{R}^m$ is the control vector. The Jacobian linearization at the origin results in a linear model given by $\dot{x} = Ax + Bu$, where $A = \dfrac{\partial f(x)}{\partial x}$ at $x = \mathbf{0}$ and $B = g(\mathbf{0})$. The pair (A, B) must be controllable [44] at the origin.

(f) The system has unstable zero dynamics [44] at the origin. Systems with actuated and unactuated shape variables whose system matrices are shown in Eq. 4.6–4.8 will have unstable zero dynamics at the origin if
$\dfrac{\partial\left(M_{s_u s_u}^{-1}(q_s) G_{s_u}(q_s)\right)}{\partial q_{s_u}} < \mathbf{0} \in \mathbb{R}^{n_{s_u} \times n_{s_u}}$ at $q_s = \mathbf{0}$.
According to [29], a system satisfying properties (e) and (f) is called a *balance system*, and this work refers to it as a *balancing system*.

(g) The system has locally strong inertial coupling [118]. Systems with actuated and unactuated shape variables whose system matrices are shown in Eq. 4.6–4.8 will have locally strong inertial coupling if $\operatorname{rank}\left(\left[M_{s_u x}(q_s), M_{s_u s_a}(q_s)\right]\right) = n - m = n_{s_u}$ in the neighborhood of the origin, and $\operatorname{rank}\left(M_{s_u x}(q_s)\right) = n_{s_u}$, *i.e.*, $M_{s_u x}(q_s)^{-1}$ exists in the neighborhood of the origin.

(h) The Jacobian of the vector of gravitational forces corresponding to the unactuated variables w.r.t. the shape variables q_s at $q_s = \mathbf{0}$ exists. Moreover, its Jacobian w.r.t. every shape set at $q_s = \mathbf{0}$ exists and is invertible.
For systems with actuated and unactuated shape variables whose system matrices are shown in Eq. 4.6–4.8, the Jacobian of $G_{s_u}(q_s) \in \mathbb{R}^{n_{s_u} \times 1}$ w.r.t. q_s, *i.e.*, $\dfrac{\partial G_{s_u}(q_s)}{\partial q_s} \in \mathbb{R}^{n_{s_u} \times n_s}$, at $q_s = \mathbf{0}$ exists and is invertible only when $n_s = n_{s_u}$. Moreover, the Jacobian of $G_{s_u}(q_s)$ w.r.t. every shape set at $q_s = \mathbf{0}$ exists and is invertible. With no loss of generality, let's assume that there is just one actuated shape set q_{s_a}. Then, both $\dfrac{\partial G_{s_u}(q_s)}{\partial q_{s_u}} \in \mathbb{R}^{n_{s_u} \times n_{s_u}}$ and

$\frac{\partial G_{s_u}(q_s)}{\partial q_{s_a}} \in \mathbb{R}^{n_{s_u} \times n_{s_a}}$ at $q_s = \mathbf{0}$ exist and are invertible.

(i) For systems with actuated and unactuated shape variables whose system matrices are shown in Eq. 4.6−4.8, the Jacobian of $M^{-1}_{s_u x}(q_s) G_{s_u}(q_s) \in \mathbb{R}^{n_{s_u} \times 1}$ w.r.t. q_s, *i.e.*, $\frac{\partial \left(M^{-1}_{s_u x}(q_s) G_{s_u}(q_s) \right)}{\partial q_s} \in \mathbb{R}^{n_{s_u} \times n_s}$, at $q_s = \mathbf{0}$ exists but is invertible only when $q_s = q_{s_u}$. Moreover, the Jacobian of $M^{-1}_{s_u x}(q_s) G_{s_u}(q_s)$ w.r.t. every shape set at $q_s = \mathbf{0}$ exists and is invertible. With no loss of generality, let's assume that there is just one actuated shape set q_{s_a}. Then, both $\frac{\partial \left(M^{-1}_{s_u x}(q_s) G_{s_u}(q_s) \right)}{\partial q_{s_u}} \in \mathbb{R}^{n_{s_u} \times n_{s_u}}$ and $\frac{\partial \left(M^{-1}_{s_u x}(q_s) G_{s_u}(q_s) \right)}{\partial q_{s_a}} \in \mathbb{R}^{n_{s_u} \times n_{s_a}}$ exist and are invertible.

The significance of properties (g)−(i) will be explained in Sec. 4.2, where they are used for designing the shape trajectory planner.

All the above listed properties are verified for the ballbot without arms and the ballbot with arms in Appendices B.1 and B.2 respectively. Apart from the ballbot, other examples of shape-accelerated balancing systems include planar and 3D cart-pole systems with unactuated lean angles, and planar balancing wheeled robots like the Segway [83] moving in a plane. The marble-maze robot is also an example of a shape-accelerated balancing system.

4.1.3 Dynamic Constraints

For underactuated systems with actuated and unactuated shape variables whose system matrices are shown in Eq. 4.6−4.8, the last $n - m$ equations of motion that correspond to their unactuated degrees of freedom given by

$$M_{s_u x}(q_s)\ddot{q}_x + M_{s_u s_a}(q_s)\ddot{q}_{s_a} + M_{s_u s_u}(q_s)\ddot{q}_{s_u} + C_{s_u s_a}(q_s, \dot{q}_s)\dot{q}_{s_a} + C_{s_u s_a}(q_s, \dot{q}_s)\dot{q}_{s_u} + G_{s_u}(q_s) = \mathbf{0} \tag{4.9}$$

can be written as:

$$\Phi(q_s, \dot{q}_s, \ddot{q}_s, \ddot{q}_x) = \mathbf{0}. \tag{4.10}$$

Equations 4.9 and 4.10 are called *second-order nonholonomic constraints*, or *dynamic constraints* because they are non-integrable [88]. They are not even partially integrable. The dynamic constraint equations in Eq. 4.10 are independent of the position and velocity of the position variables, *i.e.*, q_x and $\dot{q}_x$, but relate the acceleration of position variables $\ddot{q}_x$ to the position, velocity and acceleration of shape variables, *i.e.*, $\left(q_s, \dot{q}_s, \ddot{q}_s\right)$. It shows that non-zero shape con-

figurations result in the acceleration of position variables, and hence the name *shape-accelerated balancing systems*.

The ballbot with arms shown in Fig. 3.4(a) satisfies all properties of shape-accelerated balancing systems listed in Sec. 4.1.2. It has two actuated position configurations given by its ball angles $\theta \in \mathbb{R}^{2\times 1}$, four actuated shape configurations given by its left arm angles $\alpha^l \in \mathbb{R}^{2\times 1}$ and its right arm angles $\alpha^r \in \mathbb{R}^{2\times 1}$, and two unactuated shape configurations given by its body angles $\phi \in \mathbb{R}^{2\times 1}$. The matrices in Eq. 4.2 for the ballbot with arms are of the form:

$$M(q) = \begin{bmatrix} M_{\theta\theta} & M_{\theta\alpha^l}(q_s) & M_{\theta\alpha^r}(q_s) & M_{\theta\phi}(q_s) \\ M_{\alpha^l\theta}(q_s) & M_{\alpha^l\alpha^l}(q_s) & M_{\alpha^l\alpha^r}(q_s) & M_{\alpha^l\phi}(q_s) \\ M_{\alpha^r\theta}(q_s) & M_{\alpha^r\alpha^l}(q_s) & M_{\alpha^r\alpha^r}(q_s) & M_{\alpha^r\phi}(q_s) \\ M_{\phi\theta}(q_s) & M_{\phi\alpha^l}(q_s) & M_{\phi\alpha^r}(q_s) & M_{\phi\phi}(q_s) \end{bmatrix} \in \mathbb{R}^{8\times 8}, \tag{4.11}$$

$$C(q,\dot{q}) = \begin{bmatrix} \mathbf{0} & C_{\theta\alpha^l}(q_s,\dot{q}_s) & C_{\theta\alpha^r}(q_s,\dot{q}_s) & C_{\theta\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & C_{\alpha^l\alpha^l}(q_s,\dot{q}_s) & C_{\alpha^l\alpha^r}(q_s,\dot{q}_s) & C_{\alpha^l\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & C_{\alpha^r\alpha^l}(q_s,\dot{q}_s) & C_{\alpha^r\alpha^r}(q_s,\dot{q}_s) & C_{\alpha^r\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & C_{\phi\alpha^l}(q_s,\dot{q}_s) & C_{\phi\alpha^r}(q_s,\dot{q}_s) & C_{\phi\phi}(q_s,\dot{q}_s) \end{bmatrix} \in \mathbb{R}^{8\times 8}, \tag{4.12}$$

$$G(q) = \begin{bmatrix} \mathbf{0} \\ G_{\alpha^l}(q_s) \\ G_{\alpha^r}(q_s) \\ G_{\phi}(q_s) \end{bmatrix} \in \mathbb{R}^{8\times 1}, \tag{4.13}$$

where, each $M_{ij} \in \mathbb{R}^{2\times 2}$, each $C_{ij} \in \mathbb{R}^{2\times 2}$ and each $G_i \in \mathbb{R}^{2\times 1}$. The matrices in Eq. 4.11$-$4.13 show that the dynamics of the system are independent of both the position and the velocity of its position variables, $i.e.$, $(\theta, \dot{\theta})$. The dynamic constraint equations for the ballbot with arms are given by

$$\begin{aligned} &M_{\phi\theta}(q_s)\ddot{\theta} + M_{\phi\alpha^l}(q_s)\ddot{\alpha}^l + M_{\phi\alpha^r}(q_s)\ddot{\alpha}^r + M_{\phi\phi}(q_s)\ddot{\phi} \\ &+C_{\phi\alpha^l}(q_s,\dot{q}_s)\dot{\alpha}^l + C_{\phi\alpha^r}(q_s,\dot{q}_s)\dot{\alpha}^r + C_{\phi\phi}(q_s,\dot{q}_s)\dot{\phi} + G_{\phi}(q_s) = \mathbf{0}_{2\times 1}. \end{aligned} \tag{4.14}$$

4.2 Dynamic Constraint-based Shape Trajectory Planner

This section presents a trajectory planner that exploits the special properties of shape-accelerated balancing systems to plan shape trajectories, which when tracked will result in good approximate tracking of desired position trajectories. The dynamic constraint equations map shape configurations of the system to its acceleration in the position space, and vice versa. The trajectory planner presented in this section uses just the dynamic constraint equations to plan shape trajectories, which when tracked result in approximate tracking of desired acceleration trajectories in the position space. In order to better understand the relationship between shape configurations and accelerations in the position space, this section first analyzes a simple case where the system sticks to a constant, non-zero shape configuration, $e.g.$, the ballbot leaning at a constant body angle.

A constant, non-zero shape configuration q_s with $\dot{q}_s = \mathbf{0}$ and $\ddot{q}_s = \mathbf{0}$ reduces the dynamic constraint equations $\Phi(q_s, \dot{q}_s, \ddot{q}_s, \ddot{q}_x)$ in Eq. 4.10 to $\Phi'(q_s, \ddot{q}_x)$ given by

$$\begin{aligned} \Phi'(q_s, \ddot{q}_x) &= \Phi(q_s, \mathbf{0}, \mathbf{0}, \ddot{q}_x) \\ &= M_{s_u x}(q_s)\ddot{q}_x + G_{s_u}(q_s). \end{aligned} \tag{4.15}$$

It follows from Eq. 4.10 that

$$\Phi'(q_s, \ddot{q}_x) = \mathbf{0}. \tag{4.16}$$

The Jacobian of $\Phi'(q_s, \ddot{q}_x)$ w.r.t. $\ddot{q}_x$ at $(q_s, \ddot{q}_x) = (\mathbf{0}, \mathbf{0})$ is given by

$$\left.\frac{\partial \Phi'}{\partial \ddot{q}_x}\right|_{(q_s, \ddot{q}_x)=(\mathbf{0},\mathbf{0})} = \left. M_{s_u x}(q_s)\right|_{q_s=\mathbf{0}}. \tag{4.17}$$

By the implicit function theorem [70], if the Jacobian in Eq. 4.17 exists and is invertible then, there exists a map $\Gamma' : q_s \rightarrow \ddot{q}_x$ in the neighborhood of the origin such that $\Phi'(q_s, \Gamma'(q_s)) = \mathbf{0}$. The property (g) of shape-accelerated balancing systems listed in Sec. 4.1.2 states that $M_{s_u x}(q_s)$ exists and is invertible in the neighborhood of the origin and hence, the map Γ' exists as shown in Eq. 4.18.

$$\begin{aligned} \ddot{q}_x &= -M_{s_u x}(q_s)^{-1} G_{s_u}(q_s) \\ &= \Gamma'(q_s). \end{aligned} \tag{4.18}$$

Given any constant shape configuration, Γ' will provide the resulting acceleration in the position space. The Jacobian linearization of $\Gamma'(q_s)$ in Eq. 4.18 w.r.t. q_s at $q_s = \mathbf{0}$ gives

$$\begin{aligned} \left.\frac{\partial \ddot{q}_x}{\partial q_s}\right|_{q_s=\mathbf{0}} &= -\left.\frac{\partial\left(M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)\right)}{\partial q_s}\right|_{q_s=\mathbf{0}} \\ &= K^0_{q_s} \in \mathbb{R}^{n_x \times n_s}, \end{aligned} \tag{4.19}$$

which is a function of only system parameters, and hence it is a constant. Therefore, tracking a constant shape configuration results in a constant acceleration in the position space given by

$$\ddot{q}_x = K^0_{q_s} q_s \tag{4.20}$$

in the neighborhood of the origin.

The Jacobian of $\Phi'(q_s, \ddot{q}_x)$ in Eq. 4.15 w.r.t. q_s at $(q_s, \ddot{q}_x) = (\mathbf{0}, \mathbf{0})$ is given by

$$\left.\frac{\partial \Phi'}{\partial q_s}\right|_{(q_s, \ddot{q}_x)=(\mathbf{0},\mathbf{0})} = \left.\frac{\partial G_{s_u}(q_s)}{\partial q_s}\right|_{q_s=\mathbf{0}}. \tag{4.21}$$

By the implicit function theorem [70], if the Jacobian in Eq. 4.21 exists and is invertible then, the map Γ' in Eq. 4.18 is invertible, $i.e.$, given any acceleration in the position space, the constant shape configuration that causes it will be given by Γ'^{-1}. The property (h) of shape-accelerated balancing systems listed in Sec. 4.1.2 shows that $\dfrac{\partial G_{s_u}(q_s)}{\partial q_s} \neq \mathbf{0} \in \mathbb{R}^{n_{s_u} \times n_s}$ at $q_s = \mathbf{0}$ exists but is invertible only when all shape variables are unactuated, and the shape and position space are of equal dimensions. This implies that the map Γ' in Eq. 4.18 is invertible only when the shape and position space are of equal dimensions, and all shape variables are unactuated. Similarly, the property (i) listed in Sec. 4.1.2 shows that the linear map $K^0_{q_s}$ in Eq. 4.19 is also invertible only when the shape and position space are of equal dimensions, and all the shape variables are unactuated. For example, in the case of the ballbot without arms, both the maps Γ' and $K^0_{q_s}$ exist and are invertible.

4.2.1 Shape and position space of equal dimensions

Consider shape-accelerated balancing systems with equal number of shape and position variables, $e.g.$, the ballbot without arms. Here, all shape variables are unactuated, $i.e.$, $n_{s_a} = 0$. For such systems, the properties (h) and (i) of shape-accelerated balancing systems listed in

Sec. 4.1.2 show that both the map $\Gamma'(q_s)$ in Eq. 4.18 and the linear map $K^0_{q_s}$ in Eq. 4.19 exist and are invertible. This implies that there exists a linear map $K^0_{q_x}$ given by

$$K^0_{q_x} = (K^0_{q_s})^{-1} \in \mathbb{R}^{n_s \times n_x} \tag{4.22}$$

such that

$$q_s = K^0_{q_x} \ddot{q}_x, \tag{4.23}$$

and

$$\Phi'(K^0_{q_x} \ddot{q}_x, \ddot{q}_x) = \mathbf{0} \tag{4.24}$$

in the neighborhood of the origin.

Equations 4.20 and 4.23 show that $\ddot{q}_x$ is a constant if q_s is a constant, and vice versa. Therefore for shape-accelerated balancing systems with equal number of shape and position variables, a constant desired acceleration $\ddot{q}_x$ in the position space is achieved by tracking a constant shape configuration q_s given by Eq. 4.23.

4.2.2 High dimensional shape space

Now, let's consider shape-accelerated balancing systems with more shape variables than position variables, $e.g.$, the ballbot with arms. For such systems, the properties (h) and (i) of shape-accelerated balancing systems listed in Sec. 4.1.2 show that both the map $\Gamma'(q_s)$ in Eq. 4.18 and the linear map $K^0_{q_s}$ in Eq. 4.19 exist but are not invertible. This is obvious because the matrix $K^0_{q_s} \in \mathbb{R}^{n_x \times n_s}$ is singular with more columns than rows. This implies that there are infinite possible shape configurations that can produce the same acceleration in the position space.

However, the matrix $K^0_{q_s}$ can be split into square submatrices corresponding to each shape set. With no loss of generality, let's assume there is only one actuated shape set. Therefore, $K^0_{q_s}$ can be written as

$$K^0_{q_s} = [K^0_{q_{s_a}} K^0_{q_{s_u}}], \tag{4.25}$$

where,

$$K^0_{q_{s_a}} = -\frac{\partial\big(M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)\big)}{\partial q_{s_a}}\bigg|_{q_s=\mathbf{0}} \in \mathbb{R}^{n_x \times n_{s_a}}, \tag{4.26}$$

$$K^0_{q_{s_u}} = -\frac{\partial\big(M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)\big)}{\partial q_{s_u}}\bigg|_{q_s=\mathbf{0}} \in \mathbb{R}^{n_x \times n_{s_u}}. \tag{4.27}$$

The property (i) of shape-accelerated balancing systems listed in Sec. 4.1.2 shows that the matrices shown in Eq. 4.26 and Eq. 4.27 exist and are invertible. They represent the individual contributions of each shape set to the acceleration of the system in the position space. Therefore, these inverses provide the shape configurations, which when tracked will result in the desired acceleration in the position space. However, the individual inverses assume that the other shape sets have zero shape configurations.

Therefore, given a constant desired acceleration in the position space $\ddot{q}_x$, the constant shape configurations q_s that must be tracked to achieve $\ddot{q}_x$ can be chosen as:

$$q_s = W K^0_{q_x} \ddot{q}_x, \tag{4.28}$$

where,

$$W = \begin{bmatrix} W_{q_{s_a}} & \mathbf{0} \\ \mathbf{0} & W_{q_{s_u}} \end{bmatrix} \in \mathbb{R}^{n_s \times n_s}, \tag{4.29}$$

$$K^0_{q_x} = \begin{bmatrix} (K^0_{q_{s_a}})^{-1} \\ (K^0_{q_{s_u}})^{-1} \end{bmatrix} \in \mathbb{R}^{n_s \times n_x}. \tag{4.30}$$

The weight matrix W is chosen such that $W_{q_{s_a}} + W_{q_{s_u}} = I_{n_x}$, an $n_x \times n_x$ identity matrix. The weight matrix W allows one to relatively weigh each shape set's contribution in achieving the desired acceleration in position space.

A conventional pseudo-inverse of $K^0_{q_s}$ will return only a single set of shape configurations q_s, whereas the decoupled inverse using the weight matrix W offers more flexibility, and allows one to explore the space of infinite possible shape configurations. For example, a pure body motion or a pure arm motion or any combination of the two can be chosen for the ballbot with arms in order to achieve the same desired motion in the position space. This is particularly useful when certain physically meaningful behaviors with specific shape sets are desired like navigating a narrow corridor without arm motions.

4.2.3 Optimal Shape Trajectory Planner

Sections 4.2.1 and 4.2.2 show that the constant shape configurations needed to achieve the constant desired accelerations in the position space can be obtained from the linear maps in Eq. 4.23 and Eq. 4.28, while this section presents a shape trajectory planner that ensures optimal track-

ing of any arbitrary desired acceleration trajectory in the position space. In order to achieve a non-constant desired acceleration trajectory $\ddot{q}_x(t)$, $\dot{q}_s(t)$ and $\ddot{q}_s(t)$ will have to be non-zero.

The Jacobian of $\Phi(q_s, \dot{q}_s, \ddot{q}_s, \ddot{q}_x)$ (Eq. 4.10) w.r.t. $\ddot{q}_x$ at the origin is given by

$$\left.\frac{\partial \Phi}{\partial \ddot{q}_x}\right|_{(q_s,\dot{q}_s,\ddot{q}_s,\ddot{q}_x)=(\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0})} = \left. M_{s_u x}(q_s)\right|_{q_s=\mathbf{0}}. \tag{4.31}$$

The property (g) of shape-accelerated balancing systems listed in Sec. 4.1.2 shows that the Jacobian in Eq. 4.31 exists and is invertible. Hence, by the implicit function theorem [70], there exists a map $\Gamma : (q_s, \dot{q}_s, \ddot{q}_s) \to \ddot{q}_x$ given by

$$\begin{aligned}\Gamma(q_s, \dot{q}_s, \ddot{q}_s) = -M_{s_u x}(q_s)^{-1} \Big(& M_{s_u s_a}(q_s)\ddot{q}_{s_a} + M_{s_u s_u}(q_s)\ddot{q}_{s_u} + \\ & + C_{s_u s_a}(q_s, \dot{q}_s)\dot{q}_{s_a} + C_{s_u s_a}(q_s, \dot{q}_s)\dot{q}_{s_u} + G_{s_u}(q_s)\Big)\end{aligned} \tag{4.32}$$

such that $\Phi(q_s, \dot{q}_s, \ddot{q}_s, \Gamma(q_s, \dot{q}_s, \ddot{q}_s)) = \mathbf{0}$ in the neighborhood of the origin. However, the map Γ is not invertible even for the simple case where the shape and position space are of equal dimensions. This implies that there exists no analytical function that maps any arbitrary $\ddot{q}_x(t)$ to $(q_s(t), \dot{q}_s(t), \ddot{q}_s(t))$ such that the dynamic constraints in Eq. 4.10 are satisfied. However, any desired acceleration trajectory can be approximately tracked.

Given a desired acceleration trajectory in the position space $\ddot{q}_x^d(t)$, the proposed shape trajectory planner finds a linear map $K_{q_x} : \ddot{q}_x \to q_s$, similar to Eq. 4.23 and 4.28, such that the planned shape trajectory $q_s^p(t)$ given by

$$q_s^p(t) = W K_{q_x} \ddot{q}_x^d(t), \tag{4.33}$$

when tracked, will result in an acceleration trajectory $\ddot{q}_x^p(t)$ such that $\ddot{q}_x^p(t) \approx \ddot{q}_x^d(t)$. For the systems with shape and position space of equal dimensions, the weight matrix W is chosen to be an identity matrix, whereas for systems with high dimensional shape space, the weight matrix W is of the form shown in Eq. 4.29. The shape trajectory planning procedure is formulated as an optimization problem, where the elements of K_{q_x} in Eq. 4.33 are determined with the objective of minimizing

$$J = \left\| \ddot{q}_x^p(t) - \ddot{q}_x^d(t) \right\|_2^2, \tag{4.34}$$

where, $\ddot{q}_x^p(t)$ is obtained by substituting $q_s^p(t)$ in Eq. 4.33 and its first two derivatives into Γ in

Eq. 4.32 as follows:

$$\ddot{q}_x^p(t) = \Gamma\Big(WK_{q_x}\ddot{q}_x^d(t), WK_{q_x}\dddot{q}_x^d(t), WK_{q_x}\ddddot{q}_x^d(t)\Big). \tag{4.35}$$

This optimization can be solved using nonlinear least-squares solvers like Nelder-Mead simplex [82] and Levenberg-Marquardt [63] algorithms. For a constant desired acceleration, $K_{q_x} = K_{q_x}^0$ given in Eq. 4.22 and Eq. 4.30 will ensure optimality, whereas for any general $\ddot{q}_x^d(t)$, $K_{q_x} = K_{q_x}^0$ may not necessarily ensure optimality, but will act as a good initial guess for the optimization process. The optimality here refers to the minimum deviation of $\ddot{q}_x^p(t)$ from the desired acceleration trajectory $\ddot{q}_x^d(t)$.

The shape trajectory planning procedure presented above deals only with the tracking of a desired acceleration trajectory $\ddot{q}_x^d(t)$ and not of a desired position trajectory $q_x^d(t)$. Though the acceleration trajectory $\ddot{q}_x^p(t)$ is analytically non-integrable, it can be numerically integrated to obtain the resulting position trajectory $q_x^p(t)$ using the initial conditions of the desired acceleration trajectory $q_x^d(t)$. Therefore, with matching initial conditions, $q_x^p(t)$ approximately tracks $q_x^d(t)$ if $\ddot{q}_x^p(t) \approx \ddot{q}_x^d(t)$. The planned shape trajectory $q_s^p(t)$ and the position trajectory $q_x^p(t)$ form the feasible configuration trajectories that best approximate the desired motion in the position space.

4.2.4 Planning with Additional Shape Constraints

A system with a high dimensional shape space may need to use a subset of its shape configurations to achieve tasks other than navigation. For example, the ballbot with arms can use its arms for manipulation, which will constrain the arm angles to some specific trajectories. This section presents a variant of the shape trajectory planner that can handle these additional shape constraint trajectories, and still achieve desired motions in the position space using other available shape configurations. The shape planner assumes that there is at least one shape set available without additional constraints so as to achieve desired motions in the position space.

With no loss of generality, let's assume that there is just one actuated shape set, and it is constrained to some reference trajectory, while the unactuated shape set has no additional constraints. The objective here is to plan trajectories for the unactuated shape configurations such that they achieve the desired motion in the position space, while counteracting the effect of the additional constraints on the other shape set.

Additional constraint (ac) trajectories for the actuated shape variables $q_{s_a}^{ac}(t)$ when tracked

will result in acceleration trajectories $\ddot{q}_x^{ac}(t)$ in the position space given by

$$\ddot{q}_x^{ac}(t) = \Gamma\Big(q_s^{ac}(t), \dot{q}_s^{ac}(t), \ddot{q}_s^{ac}(t)\Big), \tag{4.36}$$

where $q_s^{ac}(t) = \left[q_{s_a}^{ac}(t), \mathbf{0}\right]^T$, *i.e.*, zero angle trajectories for the unactuated shape configurations q_{s_u}, and Γ is obtained from Eq. 4.32. In order to achieve the desired acceleration trajectory $\ddot{q}_x^d(t)$ in the position space, the unactuated shape configurations have to achieve motions that compensate for $\ddot{q}_x^{ac}(t)$, and achieve $\ddot{q}_x^d(t)$. Therefore, in this case, the planned shape trajectory $q_s^p(t)$ is chosen to be

$$q_s^p(t) = W K_{q_x} \ddot{q}_x^{net}(t), \tag{4.37}$$

where $\ddot{q}_x^{net}(t) = \ddot{q}_x^d(t) - \ddot{q}_x^{ac}(t)$ is the net desired acceleration trajectory that the planner uses for planning unactuated shape trajectories. The linear map K_{q_x} in Eq. 4.37 is obtained using the optimization procedure described in Sec. 4.2.3 with the weight matrix W chosen such that trajectories are planned only for shape variables with no additional constraints. A block diagram of the shape trajectory planner is shown in the shaded region of Fig. 4.1.

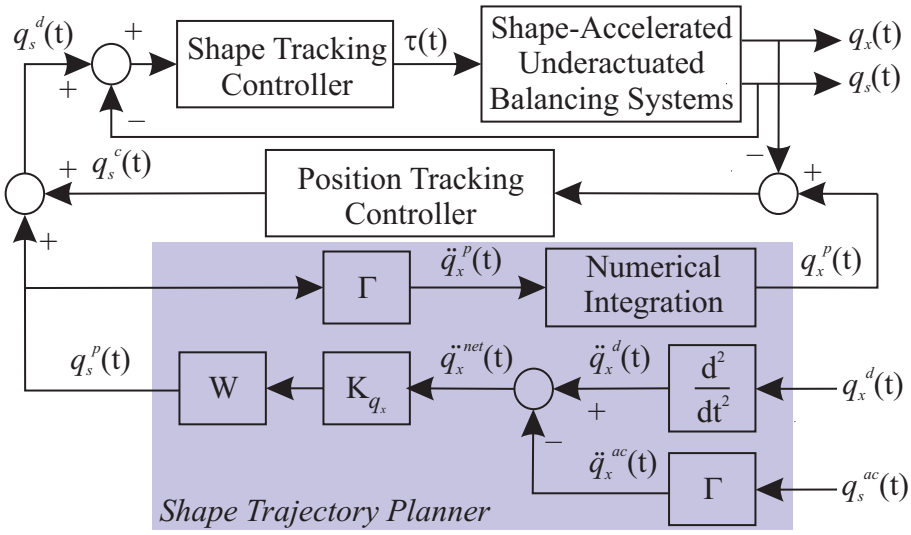


Figure 4.1: Control architecture with the shape trajectory planner (Appeared in [73]).

4.2.5 Control Architecture

The shape trajectory planner presented above assumes that there exists controllers that can accurately track the planned shape trajectories. Approximate tracking of desired position trajectories

$q_x^d(t)$ by tracking planned shape trajectories $q_s^p(t)$ is open-loop as there is no feedback on the position configurations. This open-loop procedure cannot ensure approximate tracking of desired position trajectories when the system starts at wrong initial conditions. Moreover, on real robots, there are more issues such as modeling uncertainties, unmodeled dynamics, nonlinear friction effects and noise that will inhibit a good position trajectory tracking performance. A feedback position tracking controller is used to overcome all these issues.

The feedback position tracking controller tracks planned acceleration trajectories $q_x^p(t)$, which are optimal, feasible approximations to desired position trajectories $q_x^d(t)$. The controller outputs compensation shape trajectories $q_s^c(t)$, which are added to planned shape trajectories $q_s^p(t)$ to produce desired shape trajectories $q_s^d(t)$ that are tracked by shape trajectory tracking controllers as shown in Fig. 4.1. For the ballbot with arms, the shape trajectory tracking controllers include the balancing controller and the trajectory tracking controllers for the arms, while for the ballbot without arms, the shape trajectory tracking controller includes only the balancing controller. The weight matrix W selects and relatively weighs the shape variables used for achieving desired motions in the position space.

4.2.6 Characteristics of Desired Position Trajectories

The shape trajectory planner presented in this chapter requires that the desired position trajectories $q_x^d(t)$ must be at least of differentiability class C^2, so that the desired acceleration trajectories $\ddot{q}_x^d(t)$ exist and are continuous. However, it is preferred to have $q_x^d(t)$ be of differentiability class C^4 so that the planned shape trajectories $q_s^p(t)$ and their first two derivatives ($\dot{q}_s^p(t)$, $\ddot{q}_s^p(t)$) that depend on them exist and are continuous.

Moreover, the desired position trajectories must satisfy acceleration bounds that depend on the shape configurations used to achieve these motions. The planner plans shape trajectories that are linearly proportional to desired acceleration trajectories in position space, and will fail to achieve good approximate tracking if it is outside this linear neighborhood around the origin. The nonlinear map $\Gamma'(q_s)$ in Eq. 4.18 of the ballbot with arms as a function of the body angle and the arm angle are shown in Fig. 4.2. The acceleration bounds on the desired ball position trajectories are chosen to be 2 m/s^2 for using the body angles, and 0.082 m/s^2 for using the arm angles as shown by the highlighted regions in Fig. 4.2. These acceleration bounds correspond to a maximum body angle of 10° and a maximum arm angle of 55°. The linear approximation of the nonlinear map $\Gamma'(q_s)$ works well within these bounded regions. These bounds were used for all the experimental results presented in Sec. 4.3.

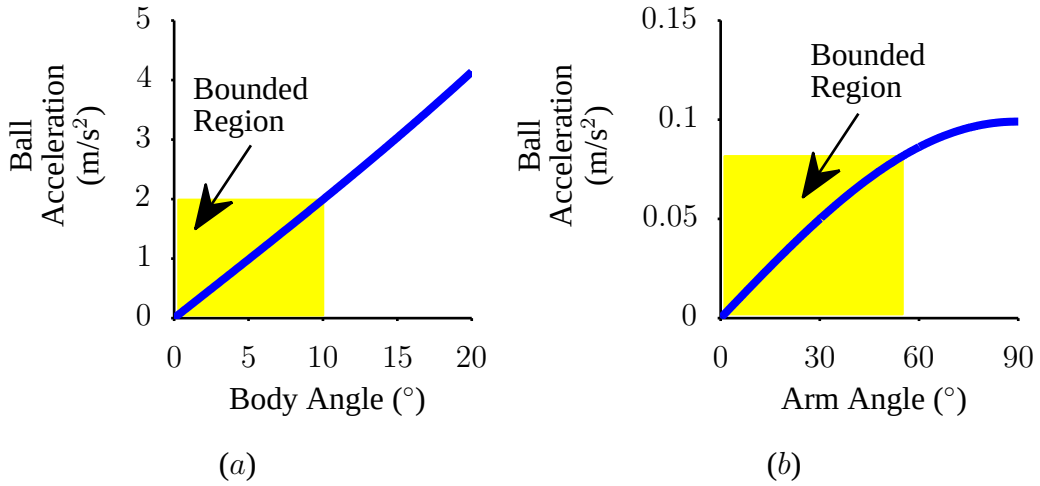


Figure 4.2: Nonlinear function of ball acceleration $vs.$ shape configuration for the ballbot with arms: (a) Body Angle; (b) Arm Angle. (Appeared in [73])

4.2.7 Choosing Weight Matrices

The shape trajectory planner assumes that a valid weight matrix W is chosen, and the conditions that determine its validity are as follows: (i) each element w_{ij} of the weight matrix W must be non-negative, $i.e.$, $w_{ij} \geq 0$; (ii) the weight matrix W must be of the form shown in Eq. 4.29; and (iii) its constituent submatrices for all shape sets must sum to an identity matrix, $e.g.$, for the ballbot with arms, $W_{\alpha^l} + W_{\alpha^r} + W_{\phi} = I_2$.

The weight matrix can also be used to account for self-collision constraints, and its elements can be chosen such that the arm motions do not collide with the body. Let's consider the case of the ballbot achieving a lateral ball motion using just its arms, where a single arm cannot produce the whole motion as it will result in collision with the body. In such a case, one arm must be used for the "acceleration-phase", while the other arm must be used for the "deceleration-phase". This can be achieved by using a different weight matrix for each phase, and such a case is experimentally demonstrated in Sec. 4.3.

4.2.8 Performance Comparison against Direct Collocation Methods

Direct collocation methods [35, 130, 131] have emerged as popular numerical techniques to generate feasible trajectories for nonlinear systems. The state and control trajectories are discretized into finite collocation points, and the trajectory generation is solved as an optimization problem subject to nonlinear constraints given by the equations of motion of the system.

Table 4.1: Performance Comparison

System (No. of states)	Computation Time (s)		Speed Factor
	PROPT	Shape Planner	
Planar cart-pole (4)	10.546	**0.230**	**46**×
Planar ballbot (4)	8.054	**0.142**	**58**×
3D ballbot without arms (8)	11.895	**0.466**	**25**×
3D ballbot with arms (16)	584.853	**8.398**	**70**×

Table 4.1 compares the performance of PROPT [104], a fast optimal control platform for MATLAB that uses direct collocation methods against that of the shape trajectory planner presented in this chapter on four different shape-accelerated balancing systems listed. The trajectory optimization was performed using the SNOPT solver [30] on PROPT. The shape trajectory planner presented in this chapter was implemented using the *lsqnonlin* function in MATLAB, which uses the Levenberg-Marquardt Algorithm (LMA) [63] for optimization. The objective was to minimize the sum of squared errors (SSE) in tracking a desired straight line position space motion of 2 m in 10 s with a functional tolerance of $<10^{-3}$ m^2. The PROPT implementation used 100 collocation points (sampling at 10 Hz), which were necessary for generating reasonably smooth trajectories.

The shape trajectory planner presented in this chapter is able to generate feasible trajectories at $25-70$ times faster speeds than PROPT on a standard Core2-Duo processor. The computation times listed are average values over 10 runs. This speed is not surprising as the optimization is performed on a much smaller parameter space compared to that of the direct collocation method. Moreover, the shape planner uses only the dynamic constraint equations, whereas PROPT uses all the equations of motion as constraints. It is to be noted that the computation times presented in Table 4.1 are for a MATLAB implementation of the shape trajectory planner. A well optimized C/C++ implementation can provide the results an order of magnitude faster, which allows real-time planning on the robot.

4.3 Experimental Results with The Ballbot

The shape trajectory planner and the control architecture presented in Sec. 4.2 were experimentally validated on the ballbot with arms shown in Fig. 3.4(a). The arms had 1 kg weights at their ends for the experiments presented in this section. The balancing controller was used to

track the desired body angle trajectories. The trajectory tracking controller for the arms used the computed torque method [71] for feedforward terms and a PID position controller for feedback control [81]. Different weight matrices were picked to select and relatively weigh the body and arm motions.

4.3.1 Pure Body Motion

Figure 4.3 shows the ballbot without arms successfully tracking a fast, straight line motion of 1.414 m in 4 s, while reaching a peak velocity of 1.18 m/s and a peak acceleration of 1 m/s^2. The planned and compensation body angle trajectories are shown in Fig. 4.4. The compensation body angle trajectory is provided by the feedback position tracking controller, and is summed with the planned body angle trajectory to produce the desired body angle trajectory that is tracked by the balancing controller. The resulting body angle trajectory and the error in tracking the desired body angle trajectory are shown in Fig. 4.5. The video of the ballbot achieiving this motion can be found in Video D.3.

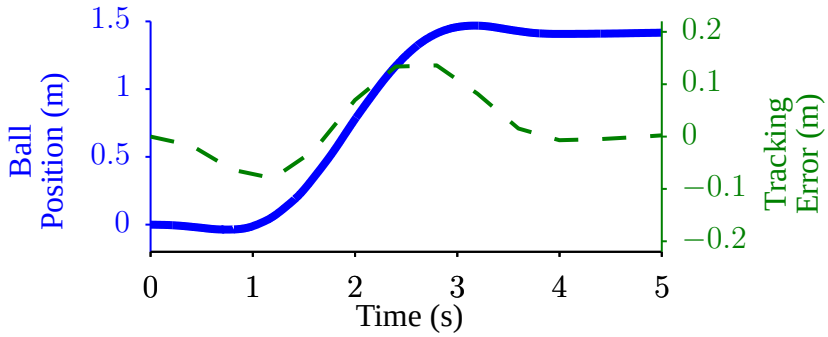


Figure 4.3: Pure Body Motion - Tracking the desired straight line motion (Appeared in [73]).

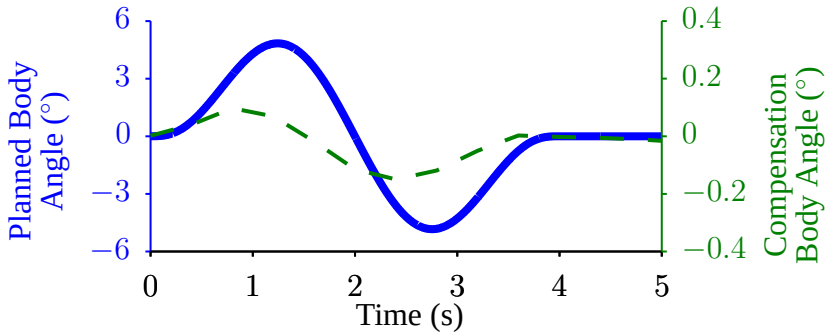


Figure 4.4: Pure Body Motion - Planned and compensation body angle trajectories for achieving the desired straight line ball motion (Appeared in [73]).

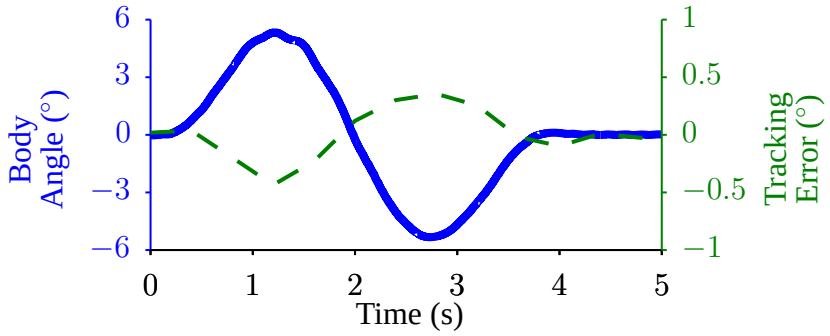


Figure 4.5: Pure Body Motion - Tracking the desired body angle trajectory for achieving the desired straight line ball motion (Appeared in [73]).

Figure 4.6 shows the ballbot with arms successfully tracking a desired straight line ball motion of 2 m using just the body angle motions. Here, the weight matrix W was chosen such that the shape trajectories were planned only in the space of body angles.

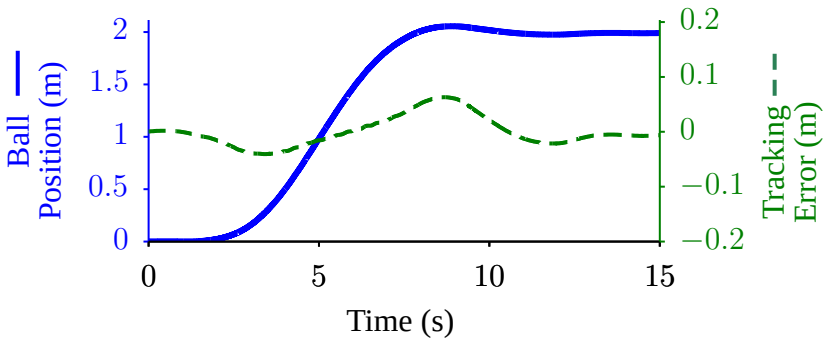


Figure 4.6: Pure Body Motion - Tracking the desired straight line motion (Appeared in [73, 81]).

The planned and compensation body angle trajectories are shown in Fig. 4.7. The compensation body angle trajectory is obtained from the feedback position trajectory tracking controller, and the small compensation angles show the effectiveness of the planned shape trajectory. The planned and compensation body angle trajectories are summed to produce the desired body angle trajectory, which is tracked by the balancing controller. Figure 4.8 shows the ballbot's resulting body angle trajectory, and the error in tracking the desired body angle trajectory by the balancing controller.

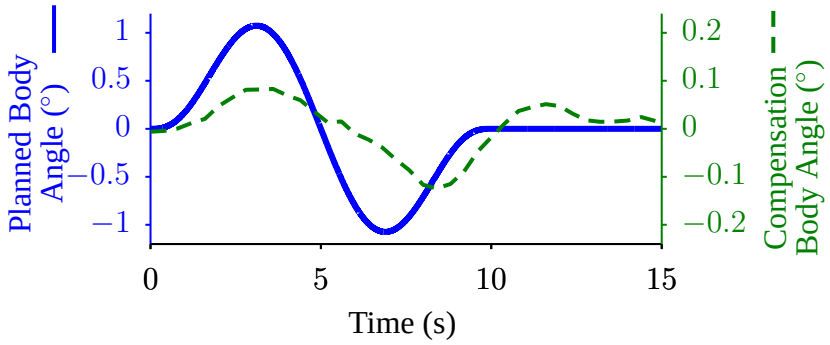


Figure 4.7: Pure Body Motion - Planned and compensation body angle trajectories for achieving the desired straight line ball motion (Appeared in [73, 81]).

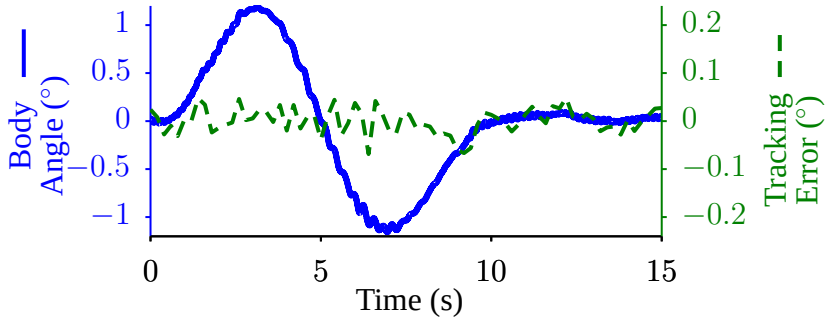


Figure 4.8: Pure Body Motion - Tracking the desired body angle trajectory for achieving the desired straight line ball motion (Appeared in [73, 81]).

The ballbot with arms tracking a curvilinear ball motion is shown in Fig. 4.9. The resulting body angle trajectories and the tracking errors are shown in Fig. 4.10 and Fig. 4.11. The compensation body angles remained within $\pm 0.15°$ for this case. The arms were maintained at zero angles for both the experiments. The videos of the ballbot with arms achieving the straight line and curvilinear motions can be found in Video D.4.

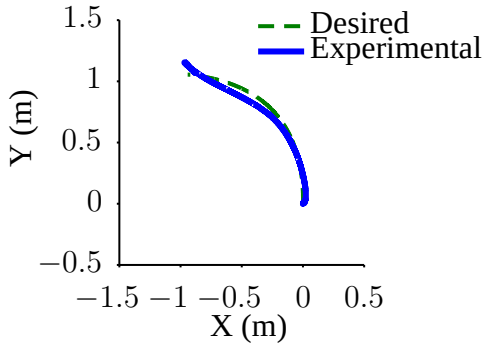


Figure 4.9: Pure Body Motion - Tracking the desired curvilinear motion (Appeared in [73, 81]).

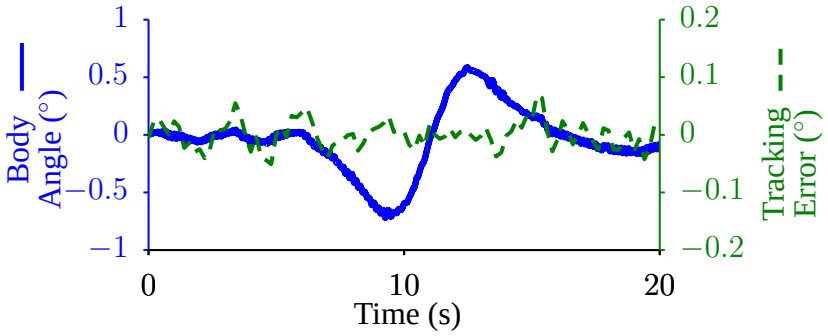


Figure 4.10: Pure Body Motion - Tracking the desired X body angle trajectory for achieving the desired curvilinear ball motion (Appeared in [73, 81]).

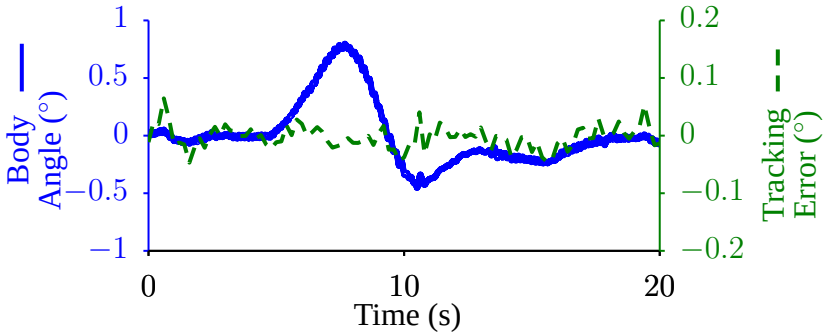


Figure 4.11: Pure Body Motion - Tracking the desired Y body angle trajectory for achieving the desired curvilinear ball motion (Appeared in [73, 81]).

4.3.2 Pure Arm Motion

This section presents experimental results of the ballbot with arms achieving desired ball motions using just the arm motions. The arms of the ballbot are lightweight hollow alumnium tubes with 1 kg massess at the ends. Figure 4.12 shows the robot tracking a desired straight line motion of 2 m in the forward direction using just the arm motions. The planned and compensation arm angle trajectories for the left arm are shown in Fig. 4.13. The desired arm trajectory tracking performance is shown in Fig. 4.14. Similar results were obtained for the right arm.

Compared to the results in Fig. 4.6, Fig. 4.12 shows that there is larger ball position tracking error while using just the arms. This is due to the relatively poor trajectory tracking performance of the arm controller as shown in Fig. 4.14, which in turn is due to some excessive backlash in the arm gears. A better arm design will significantly improve these results. The composite frames from a video of the ballbot achieving this motion is shown in Fig. 4.15, and the video can

be found in Video D.4. The balancing controller maintained the body angles at zero for these experiments.

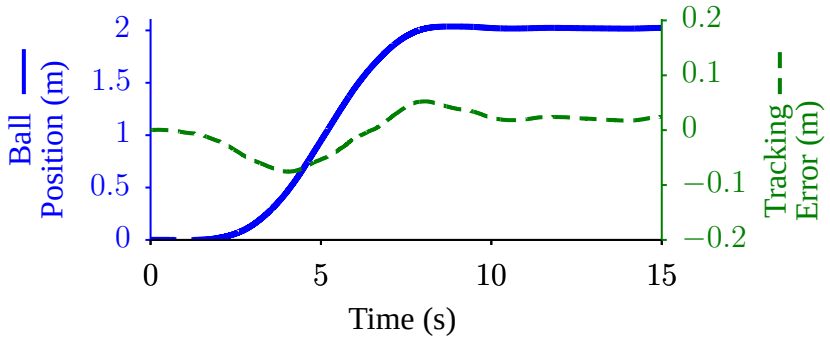


Figure 4.12: Pure Arm Motion - Tracking the desired forward straight line ball motion (Appeared in [73, 81]).

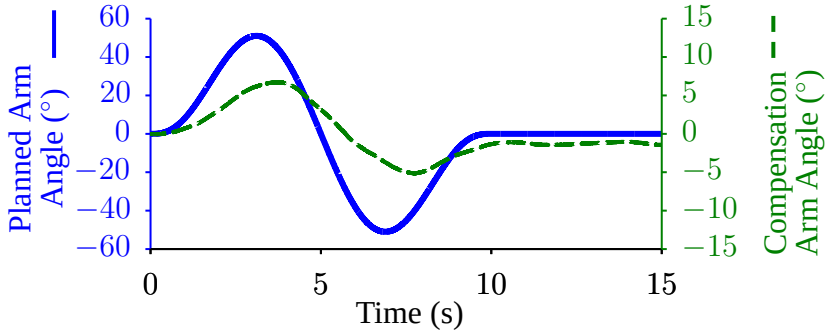


Figure 4.13: Pure Arm Motion - Planned and compensation left arm angle trajectories for achieving the desired forward ball motion (Appeared in [73, 81]).

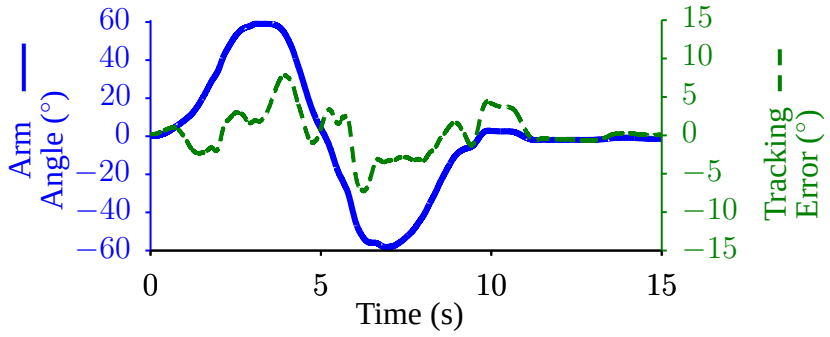


Figure 4.14: Pure Arm Motion - Tracking the desired left arm angle trajectory for achieving the desired forward ball motion (Appeared in [73, 81]).

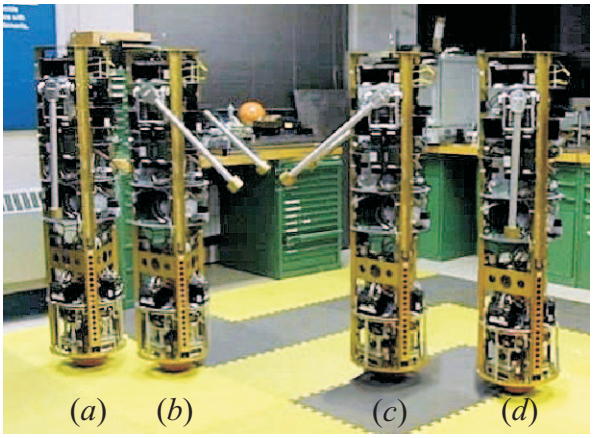


Figure 4.15: Composite frames from a video of the forward ball motion using the arms: (a) the robot starts at rest; (b) the arms move forward to accelerate; (c) the arms move backward to decelerate; and (d) the robot comes to rest. (Appeared in [73])

Figure 4.16 shows the robot tracking a desired straight line motion of 1 m in the lateral direction. Here, the arms are moved sideways, and the right arm is used to initiate the motion as shown in Fig. 4.17, whereas the left arm is used to bring the system to rest as shown in Fig. 4.18. The complete motion is not performed on a single arm in order to avoid self-collision. The weight matrix W can be chosen such that the arm motions do not collide with the body. Two different weight matrices are used for this motion, one for the "acceleration-phase" that picks the right arm, and the other for the "deceleration-phase" that picks the left arm. Figure 4.19 shows composite frames of the ballbot achieving the lateral motion using just the arms, and the video can be found in Video D.4.

For both forward and lateral motions, the compensation arm angles remained within $\pm 5^\circ$ and the balancing controller maintained the body angles within $\pm 0.05^\circ$.

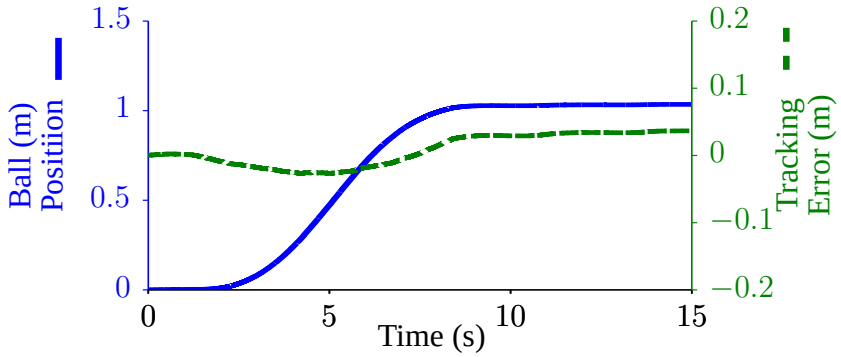


Figure 4.16: Pure Arm Motion - Tracking desired lateral ball motion (Appeared in [73, 81]).

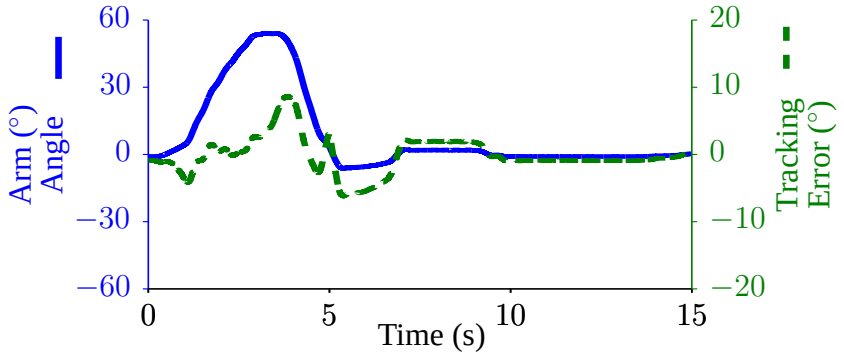


Figure 4.17: Pure Arm Motion - Tracking desired right arm angle trajectory for lateral ball motion (Appeared in [73, 81]).

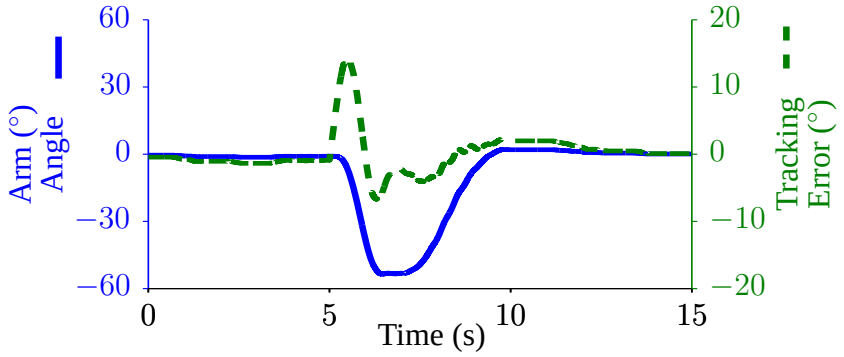


Figure 4.18: Pure Arm Motion - Tracking desired left arm angle trajectory for lateral ball motion (Appeared in [73, 81]).

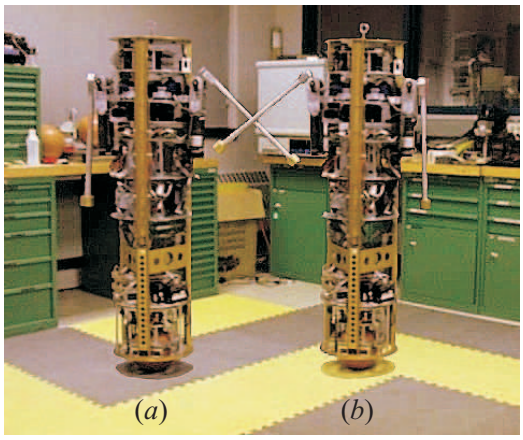


Figure 4.19: Composite frames from a video of the lateral ball motion using the arms: (a) the right arms move to accelerate; and (b) the left arms move to decelerate. (Appeared in [73])

4.3.3 Arm and Body Motion

This section presents experimental results for the case where the body and arm motions equally share (50-50) the effort of tracking a desired line ball motion of 2 m as shown in Fig. 4.20. The planned and compensation trajectories for the body angle and the right arm angle are shown in Fig. 4.21 and Fig. 4.23 respectively. The trajectory tracking performance for the body and the right arm angles are shown in Fig. 4.22 and Fig. 4.24 respectively. Similar results were obtained for the left arm. In this case, the compensation body angles remained within $\pm 0.06^\circ$ and the compensation arm angles remained within $\pm 5^\circ$. A video of the ballbot achieiving this motion can be found in Video D.4.

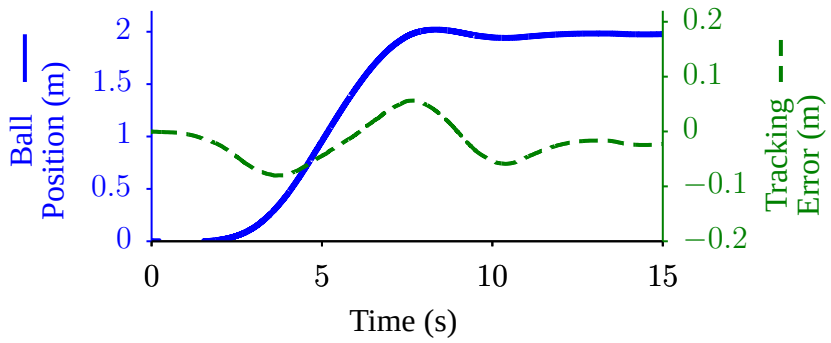


Figure 4.20: Arm and Body Motion - Tracking the desired straight line ball motion (Appeared in [73, 81]).

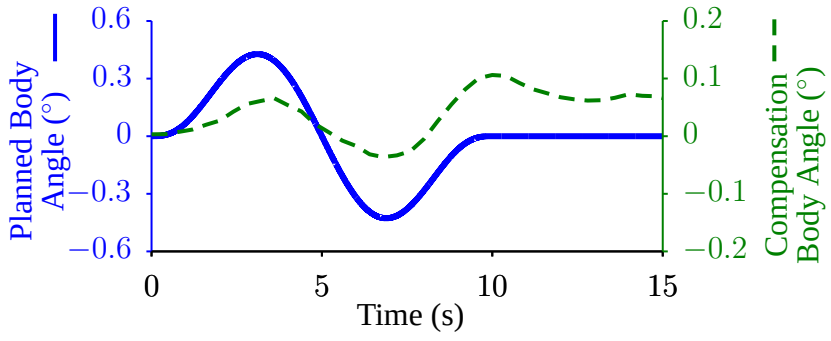


Figure 4.21: Arm and Body Motion - Planned and compensation body angle trajectories for achieving the desired straight line ball motion (Appeared in [73, 81]).

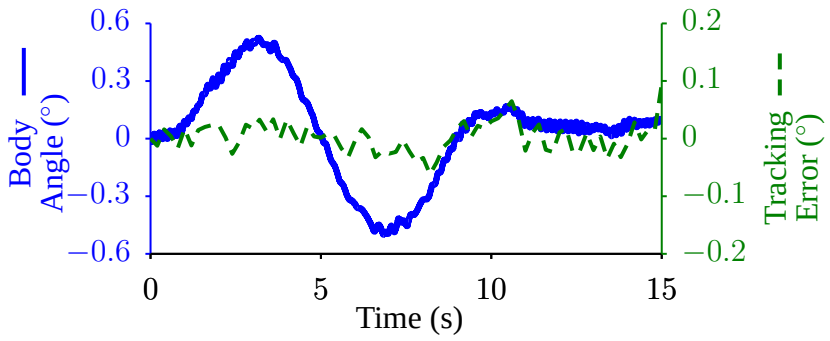


Figure 4.22: Arm and Body Motion - Tracking the desired body angle trajectory for achieving the desired straight line ball motion (Appeared in [73, 81]).

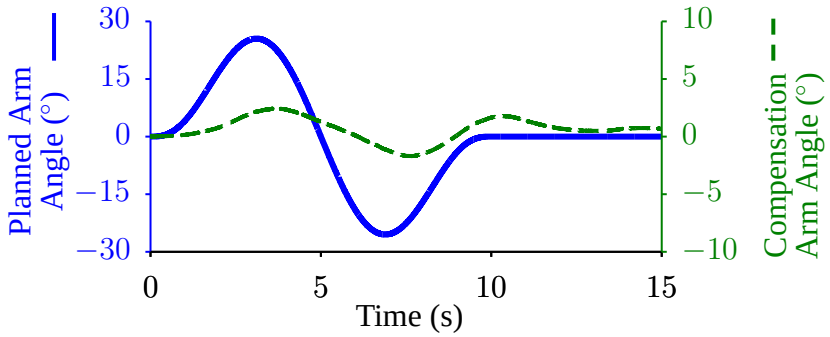


Figure 4.23: Arm and Body Motion - Planned and compensation right arm angle trajectories for achieving the desired straight line ball motion (Appeared in [73, 81]).

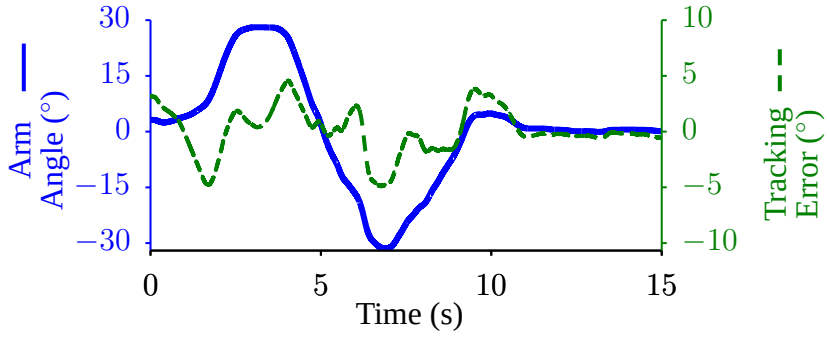


Figure 4.24: Arm and Body Motion - Tracking the desired right arm angle trajectory for achieving the desired straight line ball motion (Appeared in [73, 81]).

4.3.4 Constrained Arm Motion

The ballbot with arms was subjected to additional asymmetric constraint trajectories for the arm angles shown in Figs. 4.25−4.26. Symmetric arm motions do not result in any motion of the ball, whereas asymmetric arm motions result in the motion of the ball. Selected frames from a video of the ballbot tracking these constraint trajectories, which consist of four different goal configurations are shown in Fig. 4.27, and the video can be found in Video D.5. These arm motions were chosen to be asymmetric so that the tracking of these arm trajectories will result in the motion of the ball, if not compensated for. These arm motions were meant to emulate the robot waving its arms randomly. Since the arm angles are constrained to these trajectories, they are unavailable for shape trajectory planning and the shape trajectories were planned only in the space of body angles (Figs. 4.28−4.29) to keep the ball stationary within ± 0.04 m of its initial position as shown in Fig. 4.30.

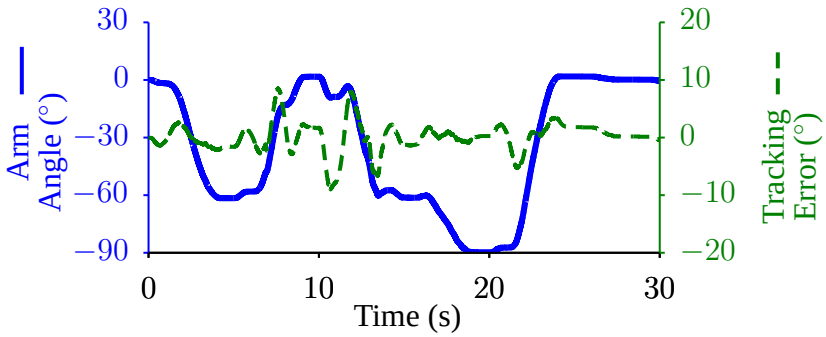


Figure 4.25: Constrained Arm Motion - Tracking the X arm angle additional constraint trajectory for the left arm (Appeared in [73]).

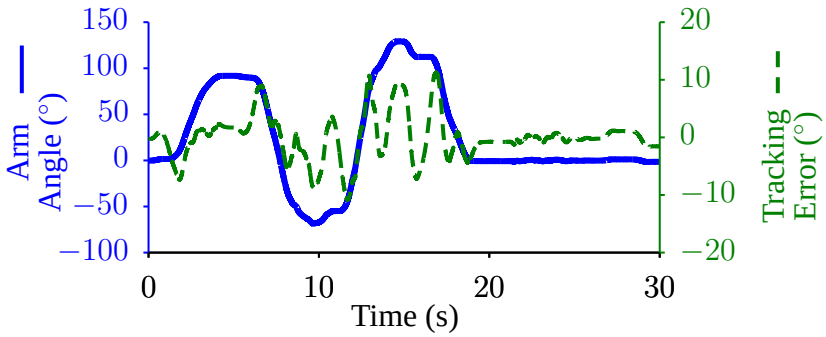


Figure 4.26: Constrained Arm Motion - Tracking the Y arm angle additional constraint trajectory for the right arm (Appeared in [73]).

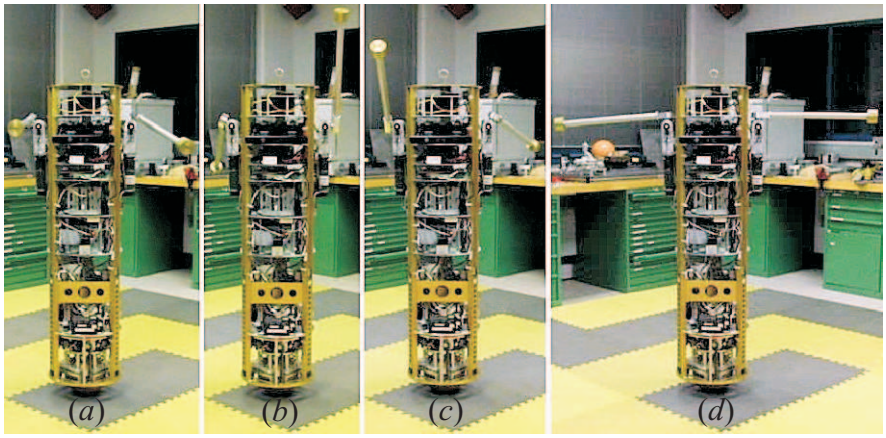


Figure 4.27: Selected frames from a video of the constrained asymmetric arm motion with four goal configurations $(a)-(d)$ (Appeared in [73]).

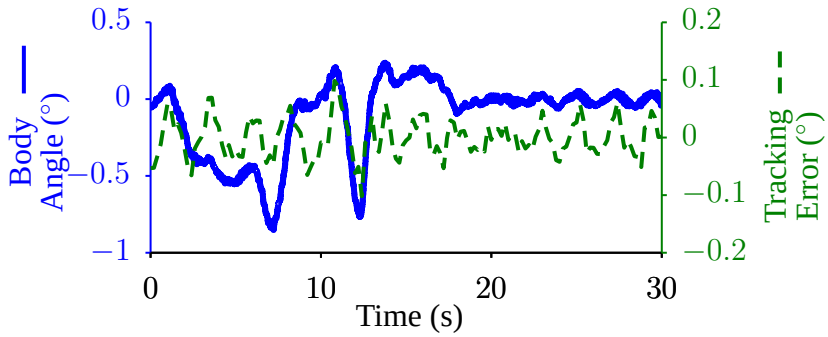


Figure 4.28: Constrained Arm Motion - Tracking the desired X body angle trajectory to achieve no ball motion (Appeared in [73]).

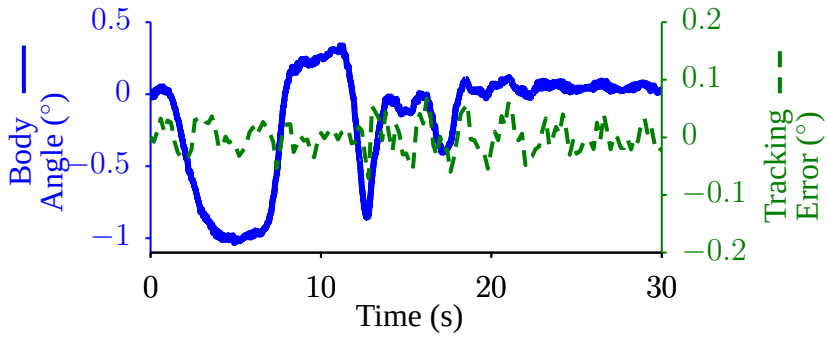


Figure 4.29: Constrained Arm Motion - Tracking the desired Y body angle trajectory to achieve no ball motion (Appeared in [73]).

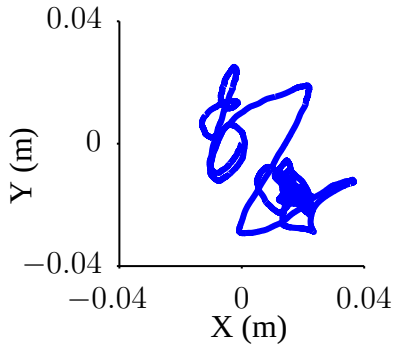


Figure 4.30: Constrained Arm Motion - Ball motion while attempting to keep it stationary (Appeared in [73]).

Figure 4.31 shows the ballbot with arms tracking a desired straight line motion of 2 m while subjected to the additional constraint of holding both its arms horizontally forward (90°) as shown in Fig. 4.32. The constraint arm trajectory consists of three motions, namely, moving the arm from 0° to 90° in the forward direction, holding it at 90° while completing the ball motion of 2 m and finally, moving the arm back from 90° to 0°. This experiment emulates the robot navigating while carrying an object.

The planned and compensation body angle trajectories are shown in Fig. 4.33 and the desired body angle tracking performance is shown in Fig. 4.34. As shown in Fig. 4.33 and 4.34, the body has to lean back to compensate for the forward held arms, and has to lean forward and backward about this angle to achieve the desired 2 m ball motion. Composite frames from a video of the ballbot performing this motion is shown in Fig. 4.35, and the video can be found in Video D.5.

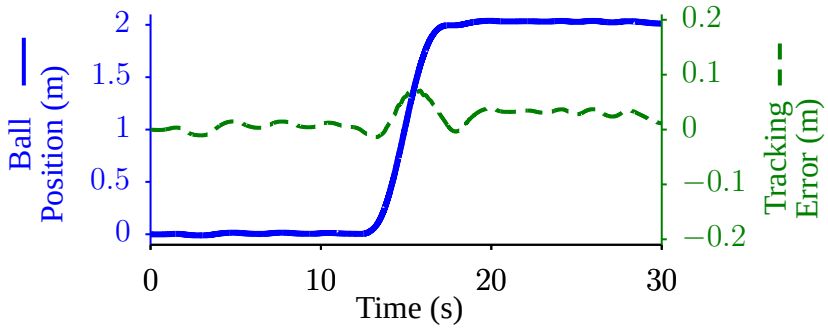


Figure 4.31: Constrained Arm Motion - Tracking the desired straight line ball motion (Appeared in [73]).

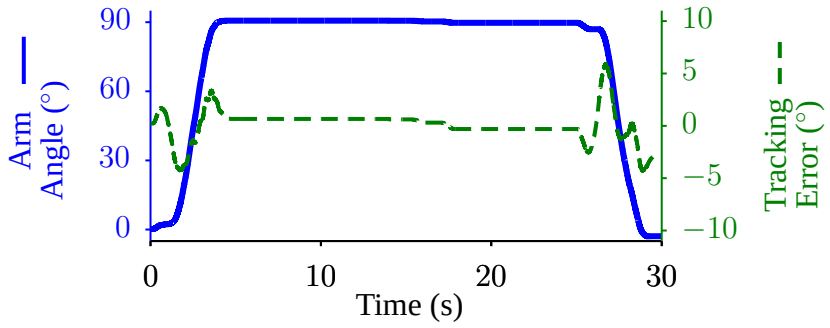


Figure 4.32: Constrained Arm Motion - Tracking the Y arm angle additional constraint trajectory for the left arm (Appeared in [73]).

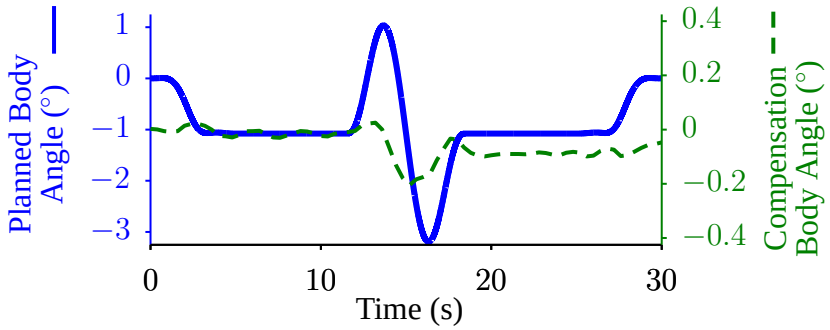


Figure 4.33: Constrained Arm Motion - Planned and compensation body angle trajectories for achieving the desired straight line ball motion (Appeared in [73]).

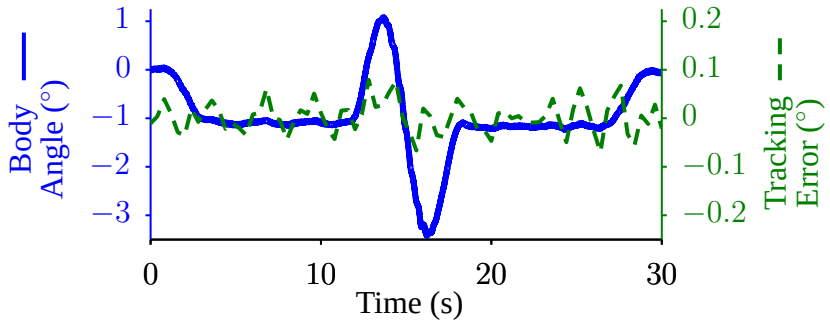


Figure 4.34: Constrained Arm Motion - Tracking the desired body angle trajectory for achieving the desired straight line ball motion (Appeared in [73]).

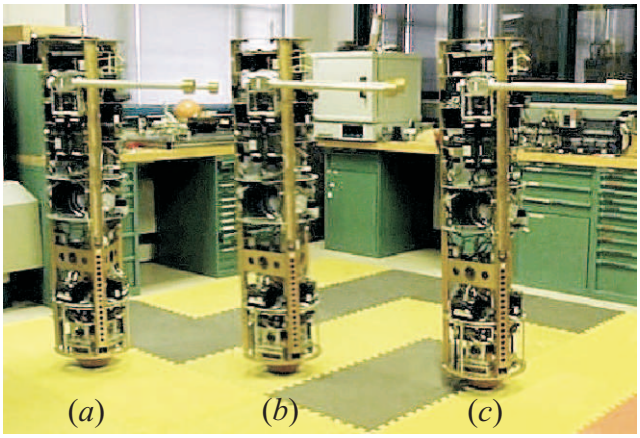


Figure 4.35: Composite frames from a video of the forward ball motion while the arms are constrained to be horizontal: (a) the body leans back to compensate for the arm constraint and then leans forward to accelerate; (b) the body leans further back to decelerate; and (c) the robot comes to rest while the body continues to lean back to compensate for the arm constraint (Appeared in [73]).

4.4 Summary

This chapter introduced shape-accelerated balancing systems as a special class of underactuated systems to which balancing mobile robots like the ballbot belong. This chapter presented a shape trajectory planning procedure for such systems that uses just the dynamic constraint equations to plan shape trajectories, which when tracked will result in optimal tracking of desired position trajectories. User-defined weight matrices were used to select and relatively weigh the contribution of different shape sets in achieving desired position space motions. The planner was also able to handle additional constraints on a subset of the shape configurations, and still plan shape space motions that will achieve desired position space motions. A feedback position trajectory tracking controller was used in parallel with the shape trajectory planner to achieve better tracking of desired position space motions. Successful experimental results on the ballbot with arms were presented. The ballbot successfully tracked the desired ball motions by tracking pure body motions, pure arm motions, their combinations, and also handled additional constraints on the arms.

The optimal shape trajectory planner presented in this chapter was shown to generate feasible state trajectories for shape-accelerated balancing systems at significantly faster speeds (25 to 70 times) than the trajectory optimization algorithms that use direct collocation methods (see

Table 4.1). The shape trajectory planning procedure exploits the special properties of shape-accelerated balancing systems and plans shape trajectories that are proportional to desired accelerations in the position space. Since this approach uses only the dynamic constraint equations and has a low-dimensional parameter space, it is significantly faster than the direct collocation methods allowing real-time generation of feasible state trajectories on-board the robot.

Chapter 5

Graceful Navigation

This chapter presents an integrated motion planning and control framework that enables balancing mobile robots like the ballbot to move gracefully and achieve desired navigation tasks (answer to RQ 2). Navigation tasks are generally posed as desired motions or states in the position space, without any specifications on shape space motions. As presented earlier in Sec. 1.4, this work defines a graceful robot motion to be any feasible robot motion in which its configuration variables' position, velocity and acceleration trajectories are continuous and bounded with low jerk.

5.1 Background

5.1.1 Decoupled Planning and Control

Traditionally, motion planning and control have been decoupled. A high-level motion planner plans a collision-free path and a low-level controller tracks it. The motion planner does not understand the capabilities and limitations of the controller, while the controller has no knowledge of the environment and the obstacles in it. This approach works well in achieving navigation tasks for kinematic mobile robots whose dynamics can be safely ignored. For balancing mobile robots like the ballbot, the shape dynamics dominate the system dynamics and cannot be ignored. The shape trajectory planning procedure [72, 73, 81] presented in Chapter 4 can be used to generate feasible state (both position and shape) trajectories that optimally achieve desired motions in the position space given by a motion planner. A decoupled approach of using a motion planner, the shape trajectory planner and the control architecture in Sec. 4.2 is capa-

ble of achieving desired navigation tasks while taking into account the dynamics of the system. Moreover, desired position space motions can be chosen such that both the resulting position and shape space motions are graceful, $i.e.$, the configuration, velocity and acceleration trajectories are continuous and bounded with low jerk. But, when subjected to large disturbances, they can result in collisions with obstacles in the environment or even drive the system unstable. This is primarily because the desired motions are achieved by tracking trajectories, and the controllers that track these trajectories have no knowledge of the obstacles in the environment.

5.1.2 Sequential Composition - A Hybrid Control Approach

Ideally, one must design control vector fields for the entire bounded state space that covers a map of the environment such that the desired navigation tasks are achieved and the obstacles are avoided. This approach will handle large disturbances as all collision-free states (including states that will result from a large disturbance) that can reach the goal will reach the goal. But designing these vector fields for high-dimensional systems that cover the entire bounded state space is a challenging task. Moreover, these design procedures are computationally expensive, and cannot be run real-time on robots.

Alternatively, one can approximately cover the bounded state space using a library of control policies generated from a small collection of pre-defined control policies using the symmetries in the system dynamics. Conner *et al.* [11] used such an approach to integrate motion planning and control to achieve navigation tasks for kinematic wheeled robots. This approach is based on *sequential composition* developed by Burridge *et al.* [10]. Given a set of control policies $\mathbb{U} = \{\Phi_1, \cdots \Phi_n\}$, each with a domain $\mathbb{D}(\Phi_i)$ and a goal set $\mathbb{G}(\Phi_i)$. A control policy Φ_1 is said to *prepare* Φ_2, denoted by $\Phi_1 \succeq \Phi_2$, if the goal of the control policy Φ_1 lies inside the domain of the control policy Φ_2, $i.e.$, $\mathbb{G}(\Phi_1) \subset \mathbb{D}(\Phi_2)$. The *prepares relationship* between control policies can be represented using cascading funnels, where one control policy leads to the other as shown in Fig. 5.1.

If the position space covered by the union of the domains of the control policies in $\mathbb{U}$ covers a map of the environment of interest, and if the union of their goal sets covers the desired navigation configurations, then any desired navigation task can be achieved using these control policies. Conner *et al.* [11] presented a semi-automated approach to generate the set of control policies $\mathbb{U}$ that fill a map of the environment from a smaller collection of control policies. A directed graph known as the *prepares graph* is generated for the set of control policies $\mathbb{U}$, where each directed edge from $\Phi_i \in \mathbb{U}$ to $\Phi_j \in \mathbb{U}$ represents the prepares relationship. If the start state S lies in

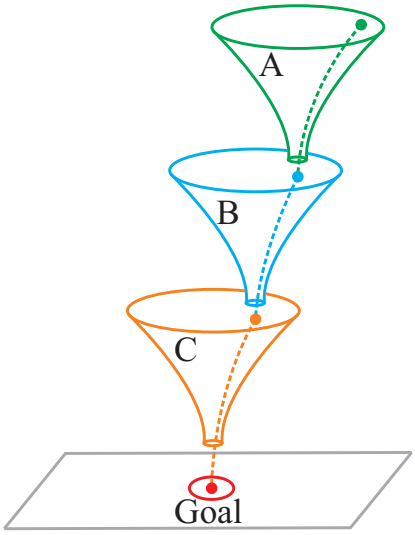


Figure 5.1: Prepares relationship represented using funnels (after Burridge *et al.* [10]).

the domain of at least one control policy, $i.e.$, $\exists i \in [1, n]$, s.t. $S \in \mathbb{D}(\Phi_i)$, and if the overall goal G lies in the goal set of at least one control policy, $i.e.$, $\exists j \in [1, n]$, s.t. $G \in \mathbb{G}(\Phi_j)$, then the navigation problem becomes a graph search problem, where the optimal sequence of control policies to reach the navigation goal can be found. The original sequential composition approach [10] defined the control policy domains in the state space of the system, whereas Conner *et al.* [11] defined the domains in the configuration space of the system. The problem of defining these domains in the high dimensional state space for balancing mobile robots like the ballbot remains a challenging problem.

In our previous work [77], the sequential composition based integrated planning and control approach was extended to balancing mobile robots like the ballbot. It used the shape trajectory planner and the control architecture presented in Chapter 4. The control architecture transforms all desired position space motions into desired shape space motions to the balancing controller, and hence allowed the domains for the control policies to be defined only in the position state space, which has lower dimensions than the whole state space. For example, both the ballbot without arms (8 states) and the ballbot with arms (16 states) have only 4 position states. Although these approaches [10, 11, 77] ensured stability and convergence of a sequential composition of control policies, and also resulted in a robust system that can navigate a map with obstacles under disturbances, the robot did not achieve graceful motion.

In summary, the decoupled planning and control using the shape space planner can produce graceful motion but is not good at handling disturbances and obstacles, whereas the sequential

composition approaches presented in [10, 11, 77] are good at handling disturbances and obstacles but do not result in graceful motion. This chapter presents an approach that combines the best of both worlds, $i.e.$, an integrated motion planning and control framework that enables balancing mobile robots like the ballbot to move gracefully and achieve desired navigation tasks while handling disturbances and obstacles.

5.1.3 Approach towards Graceful Navigation

This chapter presents an integrated motion planning and control framework based on sequential composition [10, 11, 77] that enables balancing mobile robots like the ballbot to achieve desired navigation tasks while moving gracefully. The approach presented in this chapter consists of two phases: (i) an offline controller design phase, and (ii) an online planning phase. In the offline controller design phase, controllers called *motion policies* that track feasible state trajectories called *motion primitives* are designed. A palette of motion policies is designed such that the individual motion policies result in graceful motion, and there exist combinations of motion policies that are gracefully composable. When two motion policies are gracefully composable, they guarantee graceful switching between them, thereby resulting in an overall graceful motion. In the online planning phase, a motion policy library is generated by automatically instantiating motion policies from the palette to fill a map of the environment. A motion planner plans in the space of these collision-free, gracefully composable motion policies to achieve desired navigation tasks.

5.2 Motion Policy Design

This section describes an offline procedure of designing a palette of control policies called *motion policies* for shape-accelerated balancing mobile robots like the ballbot using the optimal shape trajectory planner [72, 81] and the control architecture shown in Fig. 5.4. A motion policy Φ_i consists of a reference state trajectory called a *motion primitive* $\sigma_i(t)$, a time-varying feedback trajectory tracking control law $\phi_i(t)$, and a time-varying domain $D_i(t)$ that is verified to be asymptotically convergent [77]. All these components of a motion policy are described below.

5.2.1 Motion Primitives

Motion primitives $\sigma(t)$ are elementary, feasible state trajectories that produce motions in small domains of the position space, and they can be combined sequentially to produce more complicated trajectories. Motion primitives are feasible state trajectories, and hence by definition satisfy the constraints on the dynamics of the system. In this work, the motion primitives are defined such that they result in graceful motion, $i.e.$, their position, velocity and acceleration trajectories are continuous and bounded. Moreover, any valid sequential composition of motion primitives must also result in an overall graceful motion.

This work follows Frazzoli *et al.* [24] to define two classes of motion primitives:

(i) *Trim primitives*: Trim primitives are motion primitives that correspond to steady-state conditions and they can be arbitrarily trimmed (cut), $i.e.$, the time duration of the trajectory can be arbitrarily chosen. In this work, the trim primitives are restricted to constant position or velocity trajectories in the position space, which implies that they have zero acceleration in the position space and also have zero shape configurations.

(ii) *Maneuvers*: Maneuvers are motion primitives that start and end at steady-state conditions given by the trim primitives. Unlike trim primitives, maneuvers have fixed time durations and non-zero accelerations in the position space, which implies that they achieve non-zero shape configurations. However, maneuvers start and end at trim conditions, which correspond to zero shape configurations. Maneuvers can be any arbitrary feasible state trajectories as long as they satisfy the trim conditions.

Here, the zero shape configurations correspond to any set of shape configurations that produce zero acceleration in the position space. The motion primitives in [24] consisted of both feasible state and control trajectories, whereas the motion primitives in this work include only feasible state trajectories. Motion primitives can represent several different motions in the position space like straight line, turning, circular, S-curve or figure-8 motions.

A collection of motion primitives with a distance parameter d is defined as a *motion primitive set* $\Sigma(d)$, wherein each motion primitive produces a net Δx and Δy motion in position space such that Δx and Δy are integral multiples of the distance parameter d. Figure 5.2 presents a sample of motion primitives for the 3D ballbot model from a motion primitive set $\Sigma(d)$ with $d = 0.5$ m. It is important to note that the motion primitives for the 3D ballbot model consist of motions in 8D state space, while Fig. 5.2 shows only the corresponding 2D position space motions.

The procedure used in this work to design a motion primitive set for balancing mobile robots

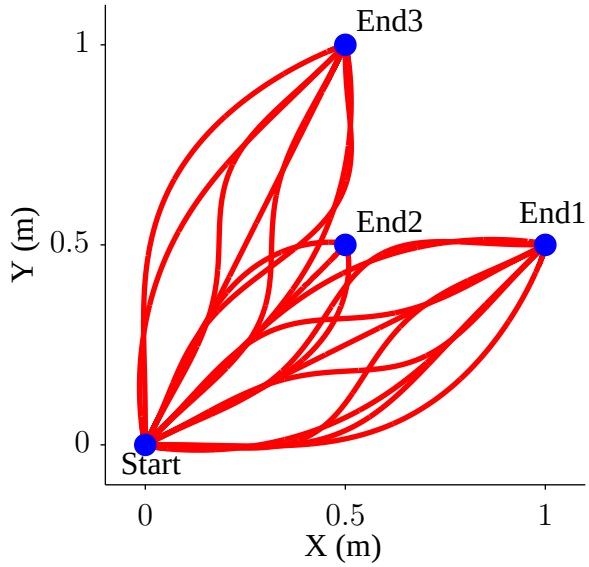


Figure 5.2: Position space motions of a sample of motion primitives for the 3D ballbot model from a motion primitive set with $d = 0.5$ m (Appeared in [79]).

like the ballbot, including the ones shown in Fig. 5.2, is described below. A number of unique, desired position space motions with a distance parameter d were chosen in the first quadrant of the XY$-$plane. The optimal shape trajectory planner [72, 81] presented in Sec. 4.2 was used to obtain the feasible position and shape trajectories that best achieve these desired position space motions. The desired trajectories in the position space were defined as nonic polynomials, $i.e.$, polynomials with degree nine, satisfying the desired boundary conditions. These boundary conditions represent trim conditions with zero acceleration in the position space. The desired trajectories were chosen such that they satisfy all characteristics of desired position trajectories presented in Sec. 4.2.6, and also satisfy all requirements of a graceful motion, $i.e.$, the position, velocity and acceleration trajectories are continuous and bounded with low jerk.

The motion primitives presented in Fig. 5.2 may strike a strong resemblance to state lattices [94, 95] and path sets [51, 52] used by the motion planners in unmanned ground vehicles. State lattices and path sets are defined as reference motions in only the position space, and the motion planners plan in the space of these reference position space motions to achieve desired navigation tasks. However, the motion primitives presented in this thesis are defined as feasible state space motions that include both position and shape space motions, and the motion planner plans in the space of motion policies, which are controllers designed around these motion primitives as will be described in Sec. 5.2.2. State lattices [94, 95] and path sets [51, 52], however, can be

used to determine the reference position space motions for generating the motion primitive sets presented in this thesis.

Each motion primitive in a motion primitive set can be rotated and translated in the position space to achieve a variety of different motions, and this process of setting the initial position and orientation of a motion primitive is called *instantiation*. This is possible because the dynamics of mobile robots are invariant to transformations of their position variables. A motion primitive set $\Sigma(d)$ is designed such that for each motion primitive $\sigma_1(t) \in \Sigma(d)$, there exists an instantiation that makes $\sigma_1(t)$ gracefully composable with at least one other motion primitive $\sigma_2(t) \in \Sigma(d)$. A motion primitive $\sigma_1(t)$ is gracefully composable with another motion primitive $\sigma_2(t)$ if and only if $\sigma_1(t_{f_i}) = \sigma_2(0)$ and $\dot{\sigma}_1(t_{f_1}) = \dot{\sigma}_2(0)$, *i.e.*, the final position, velocity and acceleration of σ_1 match the initial position, velocity and acceleration of σ_2. Figure 5.3 shows an example motion in the position space resulting from a graceful composition of appropriately instantiated motion primitives from a motion primitive set $\Sigma(d)$ with $d = 0.5$ m.

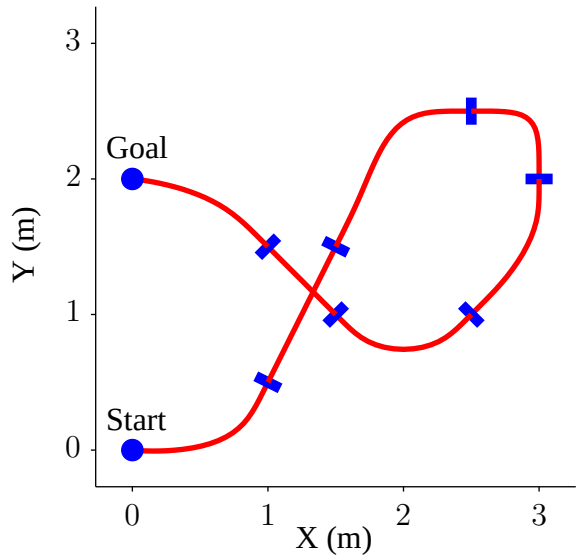


Figure 5.3: Position space motion of an example motion plan using instantiated motion primitives from Fig. 5.2. The shaded circles represent constant position trim conditions, while the bars represent constant velocity trim conditions. (Appeared in [79])

5.2.2 Motion Policies

The motion primitives presented in Sec. 5.2.1 are feasible state trajectories that satisfy the dynamic constraints of the system, and also produce graceful motion by construction. Any dynamic system requires control effort to track these feasible state trajectories. In [24], open-loop control trajectories were used as part of the motion primitives. But in a real world, especially for robots operating in human environments, where there are environment uncertainties and perturbations, one needs to use closed-loop control. This section presents *motion policies* that contain motion primitives and feedback controllers that stabilize them.

A motion policy Φ_i consists of a motion primitive $\sigma_i(t)$, a time-varying feedback tracking control law $\phi_i(t)$, and a time-varying domain $D_i(t)$, all defined for time $t \in [0, t_{f_i}]$. Since the entire motion policy execution is time parameterized, each motion policy Φ_i also contains a timer T_i that starts at zero and ticks till the duration t_{f_i} of the motion primitive $\sigma_i(t)$. A motion policy that consists of a trim primitive is called a *trim policy*, while a motion policy that consists of a maneuver is called a *maneuver policy*. Given a motion primitive set $\Sigma(d)$, one can design a *motion policy palette* $\Pi(\Sigma)$ such that each motion policy $\Phi_i \in \Pi(\Sigma)$ constitutes a motion primitive $\sigma_i \in \Sigma(d)$.

In this work, the motion policies defined for shape-accelerated balancing mobile robots like the ballbot use the control architecture shown in Fig. 5.4 (shown earlier in Fig. 4.1). It exploits the strong coupling between the dynamics of position and shape variables to achieve desired motions in the position space. The ability of this control architecture to successfully track desired motions in the position space has been experimentally verified on the ballbot [78, 81]. Since the control architecture of the motion policy achieves motions in the position space by controlling shape space motions, the motion policy domain definitions are restricted to the 4D position state space, $i.e., (x, y, \dot{x}, \dot{y})$.

The time-varying domains $D(t)$ are defined as 4D hyper-ellipsoids centered around the time-varying desired position states of the motion primitives $\sigma(t)$. Each time-varying domain $D(t)$ has a start domain $\mathbb{S} = D(0)$ and a goal domain $\mathbb{G} = D(t_f)$. Each domain $D(t)$ is verified to be asymptotically convergent, similar to the ones defined in [77]. This implies that each domain $D(t)$ has another domain $D'(t)$ defined such that $D(t) \subset D'(t)\ \forall t \in [0, t_f]$, and any position state trajectory starting in $\mathbb{S}$ will remain in $D'(t)$ until it reaches $\mathbb{G}\ \forall t \in [0, t_f]$. The overall domains for a motion policy are given by $\mathbb{D} = \bigcup_{t=0}^{t_f} D(t)$ and $\mathbb{D}' = \bigcup_{t=0}^{t_f} D'(t)$. The domain $\mathbb{D}'$ is used for checking collisions with obstacles in the environment and hence, their geometric

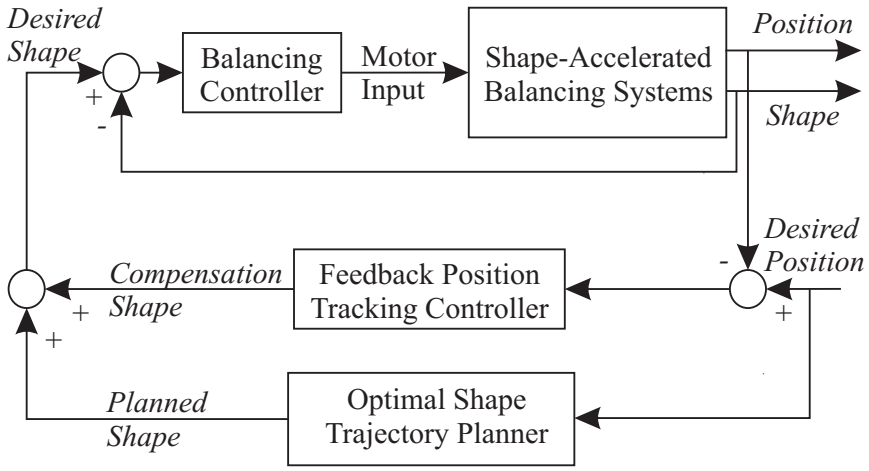


Figure 5.4: The control architecture (Appeared in [78, 79]).

definitions make it easier to verify the validity of their motion policies. Figure 5.5 shows the XY projection of a sample motion policy domain for the ballbot.

The motion policy domains presented here were verified to be asymptotically convergent [77] by simulation. The 3D ballbot model along with the control architecture in Fig. 5.4 was simulated from finitely many states in the start domain $\mathbb{S}_i$ of each motion policy Φ_i. The closed-loop position state trajectory from each of these start states was verified to remain inside the domain $D'_i(t)\ \forall t \in [0, t_f]$, and was also verified to reach the goal domain $\mathbb{G}_i$ at $t = t_f$. Additionally, the resulting closed-loop shape trajectory was also verified to remain within the domain of the balancing controller that tracks it. Several system identification experiments (Sec. 3.4) were conducted on the ballbot to estimate its system parameters such that the dynamics of the model better match the real ballbot dynamics. When a motion policy Φ_i is deployed on a map of the environment, the verification guarantees that the resulting closed-loop motion of the system in the position state space will remain within its domain $\mathbb{D}'_i$. Hence, if the domain $\mathbb{D}'_i$ is collision-free, then the motion policy Φ_i is guaranteed to produce a collision-free closed-loop motion of the system.

The process of deploying a motion policy on a map is called instantiation, just like in the case of a motion primitive. A motion policy instantiation involves the instantiation of its motion primitive and its feedback control law. The condition for two motion primitives to be gracefully composable was presented in Sec. 5.2.1. Here, a time-varying feedback control law $\phi_1(t)$ is defined to be gracefully composable with another time-varying feedback control law $\phi_2(t)$ if $\phi_1(t_{f_1}) = \phi_2(0)$, *i.e.*, the final control law of ϕ_1 matches the initial control law of ϕ_2. These

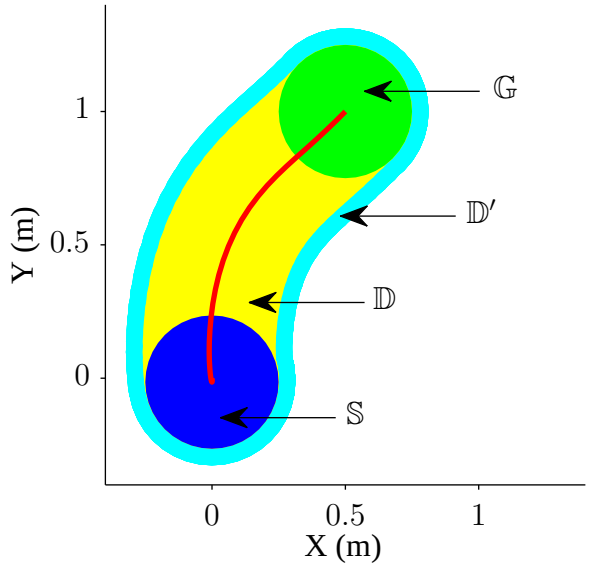


Figure 5.5: XY projection of the domain of a sample motion policy designed for the 3D ballbot model (Appeared in [78, 79]).

conditions will be used in Sec. 5.2.3 to define the graceful composition of motion policies.

5.2.3 Gracefully Prepares Relationship

This section introduces the *gracefully prepares relationship* that guarantees graceful sequential composition of two motion policies. A motion policy Φ_1 is said to *gracefully prepare* Φ_2, denoted by $\Phi_1 \succeq_G \Phi_2$, if and only if all the conditions listed below are satisfied.

(i) The goal domain of the motion policy Φ_1 is contained in the start domain of the motion policy Φ_2, *i.e.*, $\mathbb{G}_1 \subset \mathbb{S}_2$.

(ii) The motion primitive $\sigma_1(t)$ of the motion policy Φ_1 is gracefully composable with the motion primitive $\sigma_2(t)$ of the motion policy Φ_2, *i.e.*, $\sigma_1(t_{f_1}) = \sigma_2(0)$ and $\dot{\sigma}_1(t_{f_1}) = \dot{\sigma}_2(0)$, which ensures that the overall reference position, velocity and acceleration trajectories are continuous. A motion primitive $\sigma(t)$ is a state trajectory, which includes position and velocity trajectories. Therefore, its derivative $\dot{\sigma}(t)$ includes velocity and acceleration trajectories.

(iii) The time-varying feedback control law $\phi_1(t)$ of the motion policy Φ_1 is gracefully composable with the feedback control law $\phi_2(t)$ of the motion policy Φ_2, *i.e.*, $\phi_1(t_{f_1}) = \phi_2(0)$, which ensures that the overall control trajectory is continuous.

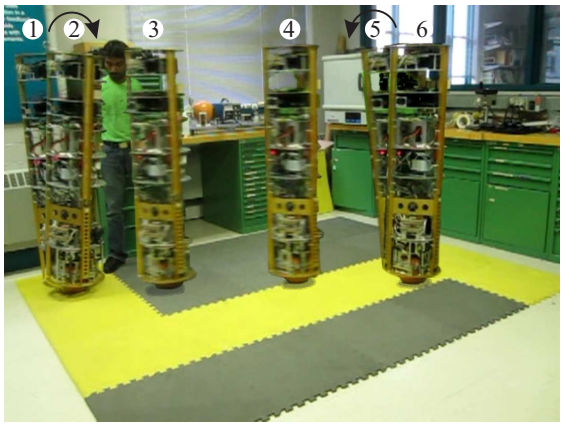
The first condition satisfies the prepares relationship [10], while the next two conditions reduce it to a gracefully prepares relationship. Hence, a gracefully prepares relationship is by definition a prepares relationship, $i.e.$, $\Phi_1 \succeq_G \Phi_2 \Rightarrow \Phi_1 \succeq \Phi_2$, but not vice-versa. Since the reference position, velocity and acceleration trajectories along with the control trajectory are continuous, the resulting closed-loop motion of the system under the action of a sequence of gracefully composable motion policies is graceful.

A good candidate for a motion policy palette $\Pi(\Sigma)$ satisfies the design features listed below.

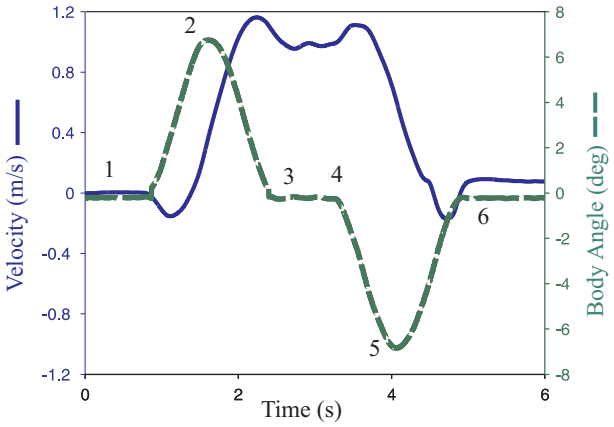
(i) For each motion primitive $\sigma_1 \in \Sigma(d)$, at least one other motion primitive $\sigma_2 \in \Sigma(d)$ is designed such that σ_1 is gracefully composable with σ_2.

(ii) For each pair of motion primitives $\sigma_1, \sigma_2 \in \Sigma(d)$ where σ_1 is gracefully composable with σ_2, their corresponding motion policies $\Phi_1, \Phi_2 \in \Pi(\Sigma)$ are designed such that Φ_1 gracefully prepares Φ_2.

Figures 5.6 and 5.7 show the experimental results of the ballbot achieving fast, graceful motions while switching between gracefully composable motion policies. The ballbot achieves a fast, graceful straight line motion in Fig. 5.6 that is composed of three different motions. The ballbot achieved a peak velocity of 1.16 m/s, a peak acceleration of 1.1 m/s^2, and a maximum lean of 6.75° in the plane of motion. In Fig. 5.7, the ballbot makes three sharp left turns by gracefully switching between nine gracefully composable motion policies. The videos of the ballbot performing both these motions can be found in Video D.6.

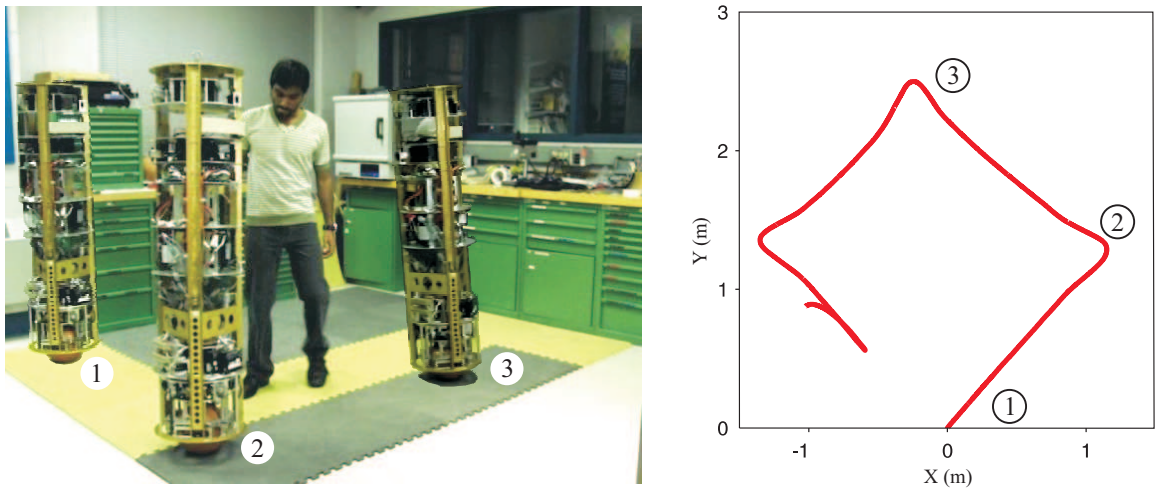


(a)



(b)

Figure 5.6: Fast straight line motion: (a) composite frames from a video, (b) plot of body angle and velocity $vs.$ time in the plane of motion.



(*a*) (*b*)

Figure 5.7: Sharp turning motion: (*a*) composite frames from a video, (*b*) plot of the motion tracked on the floor.

The sequence of gracefully composable motion policies used for the experimental results shown in Fig. 5.6 and Fig. 5.7 were manually chosen, and the switching operation was also performed manually. Section 5.3 will present automated approaches towards autonomous planning in the space of gracefully composable motion policies to achieve desired navigation tasks. It will also present a hybrid control architecture that successfully executes the motion plan.

5.3 Integrated Motion Planning and Control

An offline procedure to design a palette of gracefully composable motion policies was presented in Sec. 5.2. This section presents the online procedures that run on-board the robot to achieve graceful navigation using this palette of gracefully composable motion policies. An automatic motion policy instantiation procedure is presented, which uses the motion policy palette to generate a library of instantiated motion policies whose domains fill an obstacle-free map of the environment. A motion planner that plans in the space of these gracefully composable motion policies, and a hybrid control architecture that executes a generated motion plan are presented. A dynamic replanning algorithm that actively replans in the space of gracefully composable motion policies to avoid dynamic obstacles is also presented.

5.3.1 Automatic Instantiation of Motion Policies

Here, an automatic instantiation procedure that uniformly distributes motion policies both in position and in orientation on a 2D map of the environment is presented. Given a map $\mathbb{M}$ and a motion policy palette $\Pi(\Sigma)$, every motion policy $\Phi_i \in \Pi(\Sigma)$ is instantiated at different instantiation points that uniformly discretize the map $\mathbb{M}$. These instantiation points are separated by a distance d along both X and Y directions, where d is the distance parameter of the motion primitive set $\Sigma(d)$. At every instantiation point on the map $\mathbb{M}$, the motion policies are also instantiated in different orientations. The uniform discretization in orientation space depends on the position space motions produced by motion policies in the motion policy palette. The motion policies presented here produce motions in the first quadrant of the position space as shown in Fig. 5.2, and hence the orientation spacing is chosen to be 90° so that their instantiations can cover all four quadrants of the position space.

The automatic instantiation procedure generates a motion policy library $\mathbb{L}(\Pi, \mathbb{M})$, which is a collection of valid instantiations of motion policies from the motion policy palette $\Pi(\Sigma)$ on a map $\mathbb{M}$ of the environment. A subset of valid instantiated motion policies from a motion policy library on a map with static obstacles is shown in Fig. 5.8. An instantiated motion policy Φ_i is considered valid if and only if the following conditions are satisfied:

(i) The motion policy domains $\mathbb{D}_i$ and $\mathbb{D}'_i$ must be obstacle-free in order to guarantee that the closed-loop motion resulting from the execution of the motion policy will remain collision-free.

(ii) For motion policies that start at non-stationary trim conditions, the adjacent cell in the direction opposite to its motion must be obstacle-free. Such motion policies, if included in the motion policy library, will become orphan motion policies as other valid motion policies cannot prepare them.

(iii) For motion policies that end at non-stationary trim conditions, the adjacent cell in the direction of its motion must be obstacle-free. Such motion policies, if included in the motion policy library, will result in collisions as they cannot prepare valid motion policies.

The percentage of the bounded position state space covered by the start domains of the motion policies in the motion policy library represents the coverage of the motion policy library. The coverage percentage is calculated by uniformly sampling the bounded position state space and verifying its existence in the union of the start domains of the motion policies in the motion policy library. If the desired coverage is not achieved then the grid spacing for the instantiation points

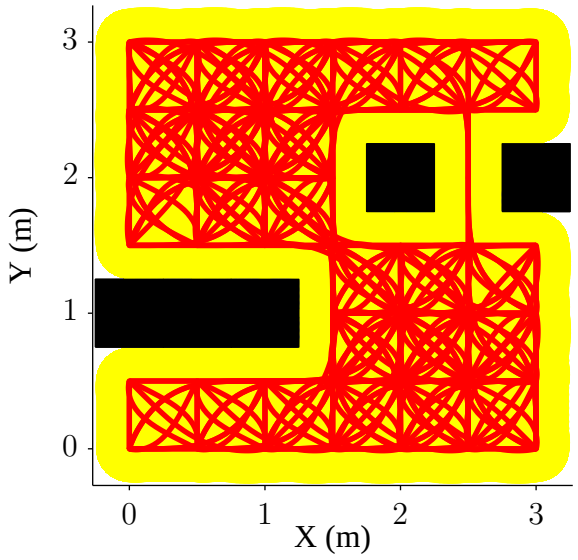


Figure 5.8: A subset of motion policies from a motion policy library $\mathbb{L}(\Pi, \mathbb{M})$, instantiated from the motion policy palette $\Pi(\Sigma)$ with the motion primitive set $\Sigma(d)$ shown in Fig. 5.2. The lines represent position space motions of the motion primitives, while the shaded regions show 2D projections of the 4D motion policy domains, including their outer domains. (Appeared in [79])

is halved, and the automatic instantiation procedure is repeated until either the desired coverage or the maximum number of such iterations is achieved. If 100% coverage is not achieved, then it indicates that there are certain states within the position state space that cannot be handled by the motion policies in the motion policy library. These position states are not reached under normal circumstances, but can be reached when the robot is subjected to large disturbances. A backup emergency controller is used to handle such cases. In this work, the ballbot switches to a simple balancing mode when such a situation is encountered.

5.3.2 Planning in Motion Policy Space

Given a map $\mathbb{M}$ and a motion policy palette $\Pi(\Sigma)$, the automatic instantiation procedure presented in Sec. 5.3.1 generates a motion policy library $\mathbb{L}(\Pi, \mathbb{M})$. The gracefully prepares relationship between every pair of motion policies (Φ_i,Φ_j) in the motion policy library $\mathbb{L}(\Pi, \mathbb{M})$ is verified using the conditions presented in Sec. 5.2.3, and a directed graph called the *gracefully prepares graph* $\Omega(\mathbb{L})$ is generated, an example of which is shown in Fig. 5.9. Each node in $\Omega(\mathbb{L})$ represents a valid instantiated motion policy $\Phi_i \in \mathbb{L}$, and each directed edge from Φ_i to Φ_j

represents the gracefully prepares relationship, $i.e.$, $\Phi_i \succeq_G \Phi_j$. The gracefully prepares graph is similar to the prepares graph presented in [11], but differs from the fact that the edges represent the gracefully prepares relationship and not just the prepares relationship as explained in Sec. 5.2.3. Unlike in the prepares graph, the switching between motion policies in the gracefully prepares graph is guaranteed to result in overall graceful motion. Therefore, the gracefully prepares graph $\Omega(\mathbb{L})$ contains all possible graceful motions that the robot can achieve on the map $\mathbb{M}$ using motion policies in the motion policy library $\mathbb{L}(\Pi, \mathbb{M})$.

This work assumes that any navigation task can be formulated as a motion between trim motion policies, $i.e.$, motion policies with trim motion primitives, and hence any navigation task can be formulated as a graph search problem on the gracefully prepares graph. This assumption is valid since any navigation task can be formulated as either a point-point motion or a surveillance motion or a combination of the two. A point-point motion can be formulated as a motion between trim motion policies that have constant position trajectories as trim primitives, while a surveillance motion can be formulated as a motion between trim motion policies that have constant velocity trajectories as trim primitives.

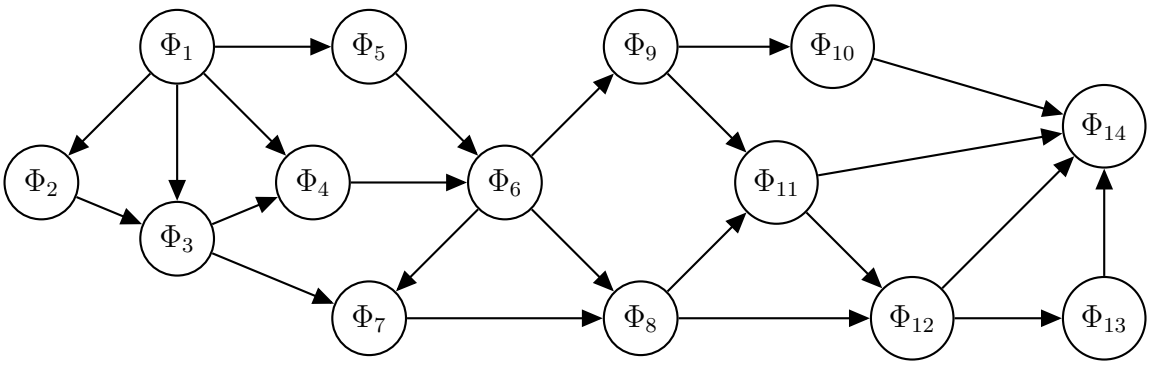


Figure 5.9: An example gracefully prepares graph.

Traditionally, graph search algorithms have been used to plan in the space of discrete cells or paths. But, in this work, graph search algorithms are used to plan in the space of gracefully composable motion policies, $i.e.$, controllers. The graph search algorithms now provide a motion plan that consists of a sequence of gracefully composable motion policies that achieve the overall navigation task. In this work, the Dijsktra's algorithm [17] shown in Algorithm 5.1 is used to solve the single-goal optimal navigation problem. The candidates for the optimality criterion include fastest time and shortest path. Unlike other heuristic-based graph search algorithms like A^* [36], the Dijsktra's algorithm uses a dynamic programming approach and optimizes over the

actual cost funtion without the use of any heuristics.

Algorithm 5.1: Single-Goal Optimal Motion Policy Tree using Dijkstra's Algorithm

input : Gracefully Prepares Graph Ω
Goal Motion Policy Node G
output: Optimal Motion Policy Tree Γ

begin
  $\Gamma \leftarrow \emptyset$
  foreach $i \in$ Node(Ω) **do**
    Cost2Goal(i) $\leftarrow \infty$
    Next2Goal(i) $\leftarrow \emptyset$
    $\Gamma \leftarrow \Gamma \cup \big(i$, Cost2Goal($i$), Next2Goal($i$)$\big)$
  end
  Cost2Goal(G) $\leftarrow 0$
  $Q \leftarrow \Gamma$
  while $Q \neq \emptyset$ **do**
    $j \leftarrow$ MinCostNode(Q)
    if Cost2Goal(j) = ∞ **then**
      break
    end
    $Q \leftarrow Q - \{j\}$
    foreach $k \in$ Parent(j) **do**
      c $\leftarrow$ Cost2Goal(j) + EdgeCost(k, j)
      if c $<$ Cost2Goal(k) **then**
        Cost2Goal(k) $\leftarrow$ c
        Next2Goal(k) $\leftarrow$ j
      end
    end
  end
end

Given a goal position state, the Euclidean distance metric is used to find the closest trim motion policy whose goal domain contains it, and its corresponding node in the gracefully prepares graph Ω forms the goal motion policy node G. Algorithm 5.1 generates a single-goal optimal motion policy tree $\Gamma(\Omega, G)$, which contains optimal paths from all motion policy nodes in the

gracefully prepares graph Ω to the goal motion policy node G. Each motion policy node i in the optimal motion policy tree $\Gamma(\Omega, G)$ contains a motion policy Φ_i, its cost to reach the goal motion policy node given by *Cost2Goal*(i), and a pointer to the next optimal motion policy node given by *Next2Goal*(i). The algorithm begins with resetting the *Cost2Goal* values and the *Next2Goal* pointers for all the motion policy nodes in the optimal policy tree Γ except for the goal motion policy node G whose *Cost2Goal* value is set to zero (*lines* $3-7$). A list of unoptimized nodes Q is created and iterated over (*lines* $10-23$). At each iteration, the motion policy node with the minimum *Cost2Goal* given by *MinCostNode*(Q) is removed from Q, and the *Cost2Goal* values and the *Next2Goal* pointers of its parents are updated (*lines* $16-22$). The function *EdgeCost*(k, j) returns the cost of switching from motion policy Φ_k to Φ_j. This iteration continues until all unoptimized nodes are optimized or all remaining motion policy nodes in Q do not have a path to the goal motion policy node G, which is indicated by infinite *Cost2Goal* (*lines* $12-14$).

The optimal motion policy tree $\Gamma(\Omega, G)$ represents all optimal graceful motions that the robot can achieve in order to reach the goal motion policy node G. However, the optimality is limited by motion policies in the motion policy library. Any motion policy node that has a path to the goal motion policy node will reach the goal motion policy node by switching between an optimal sequence of motion policies that guarantee overall graceful motion by construction. Figure 5.10 presents a subtree of a single-goal time-optimal motion policy tree obtained using Algorithm 5.1. It shows XY projections of optimal motion policy sequences along with their 4D domains from a number of different initial positions on a map with static obstacles. Since the optimal motion policy tree contains the optimal sequence of gracefully composable motion policies from all trim conditions to the goal motion policy node, replanning is unnecessary when the robot's start position changes. However, the optimal motion policy tree has to be regenerated when the goal motion policy node changes.

Since the planning is done in the space of motion policies, $i.e.,$ controllers, Algorithm 5.1 essentially generates an optimal control vector field for the bounded position state space covered by the motion policy library. The resulting closed-loop vector field optimally drives each valid position state to the goal position state. However, the optimality is limited by motion policies in the motion policy library.

This section presented a motion planner that plans in the space of gracefully composable motion policies (controllers), and explicitly accounts for both the dynamics of the system and the domains of the controllers. Each motion policy knows the exact motion it achieves in the environment and also knows that it is collision-free. Therefore, in this framework, the motion

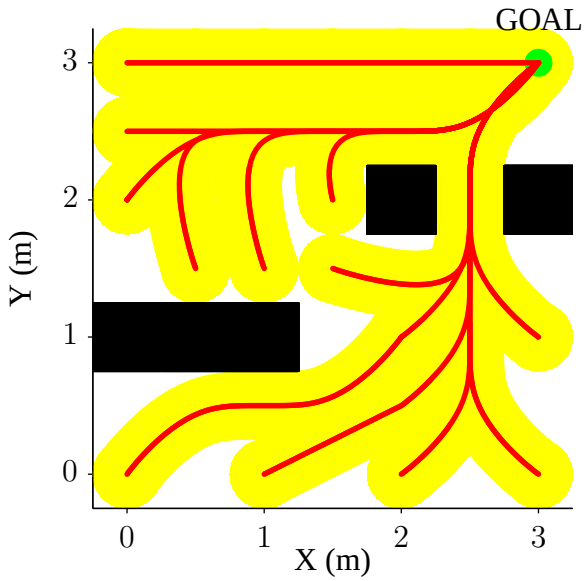


Figure 5.10: A subtree of a single-goal time-optimal motion policy tree. The lines represent position space motions of the motion primitives, while the shaded regions show 2D projections of the 4D motion policy domains, including their outer domains. (Appeared in [78, 79])

planner has knowledge of the system dynamics, and capabilities and limitations of the underlying controllers used to achieve the motion plans. The controllers, on the other hand, have knowledge of the environment constraints and also the motions that they produce, thereby, forming a truly integrated motion planning and control framework.

5.3.3 Hybrid Control

The optimal path to the goal motion policy node from any start motion policy node in the optimal motion policy tree is obtained by following its *Next2Goal* pointer until the goal motion policy node is reached. A hybrid controller is used as a master/supervisory controller to ensure successful execution of the optimal sequence of motion policies. The hybrid controller starts executing a motion policy Φ_i and resets its timer only if the robot's position state is inside its start domain $\mathbb{S}_i$. It continues executing the motion policy Φ_i only if the robot's position state lies inside its domain ${D_i}'(t)\ \forall t \in [0, t_f]$, and the motion policy execution is stopped when its timer runs out and the robot's position state reaches its goal domain $\mathbb{G}_i$. The switch to the next motion policy Φ_j happens naturally as the goal domain of Φ_i lies inside the start domain of Φ_j, *i.e.*, $\mathbb{G}_i \subset \mathbb{S}_j$ by construction of the gracefully prepares graph.

The feedback control law $\phi_i(t)$ of a motion policy Φ_i is capable of handling small disturbances and uncertainties. But when subjected to large disturbances, the robot's position state can exit the domain $D'(t)$, for some $t \in [0, t_f]$. Since the hybrid controller checks for the validity of the motion policy $\forall t \in [0, t_f]$, it detects the domain exit, stops the execution of the current motion policy, finds another motion policy Φ_k whose start domain $\mathbb{S}_k$ contains this exiting position state, and switches to the new motion policy Φ_k. If there exists a path from the new motion policy Φ_k to the goal, then the optimal motion policy tree $\Gamma(\Omega, G)$ already contains the sequence of gracefully composable motion policies that lead the motion policy Φ_k to the goal motion policy node G. If the automatic instantiation procedure guaranteed 100% coverage, then there will always exist a motion policy Φ_k in the motion policy library whose start domain will capture the robot's exiting position state. But if 100% coverage was not guaranteed and if there is no motion policy in the motion policy library whose start domain captures the robot's exiting position state, then the hybrid controller executes the backup emergency controller. In this work, the hybrid controller switches to a simple balancing mode when such a scenario is encountered. It is to be noted that the switching from the motion policy Φ_i to the new motion policy Φ_k or the backup emergency controller is discrete and is not graceful as the disturbance added to the robot that caused the domain exit is discontinuous.

5.3.4 Dynamic Replanning

While navigating human environments, it is inevitable that new static obstacles or dynamic obstacles are encountered. In such cases, a dynamic replanning procedure is necessary that will avoid motion policies that result in collisions with the new obstacles.

Figure 5.11(a) shows an example optimal motion policy tree with a goal motion policy Φ_1. The optimal motion policy tree accounts for the known static obstacles in the environment, and hence each motion policy Φ_i represents a collision-free motion policy in the known environment. However, the introduction of new static and dynamic obstacles can invalidate some of these motion policies, for example, the motion policy Φ_3 in Fig. 5.11(b) is invalidated by a new obstacle. In such cases, the optimal motion policy tree must be updated to avoid these invalid motion policies as shown in Fig. 5.11(c).

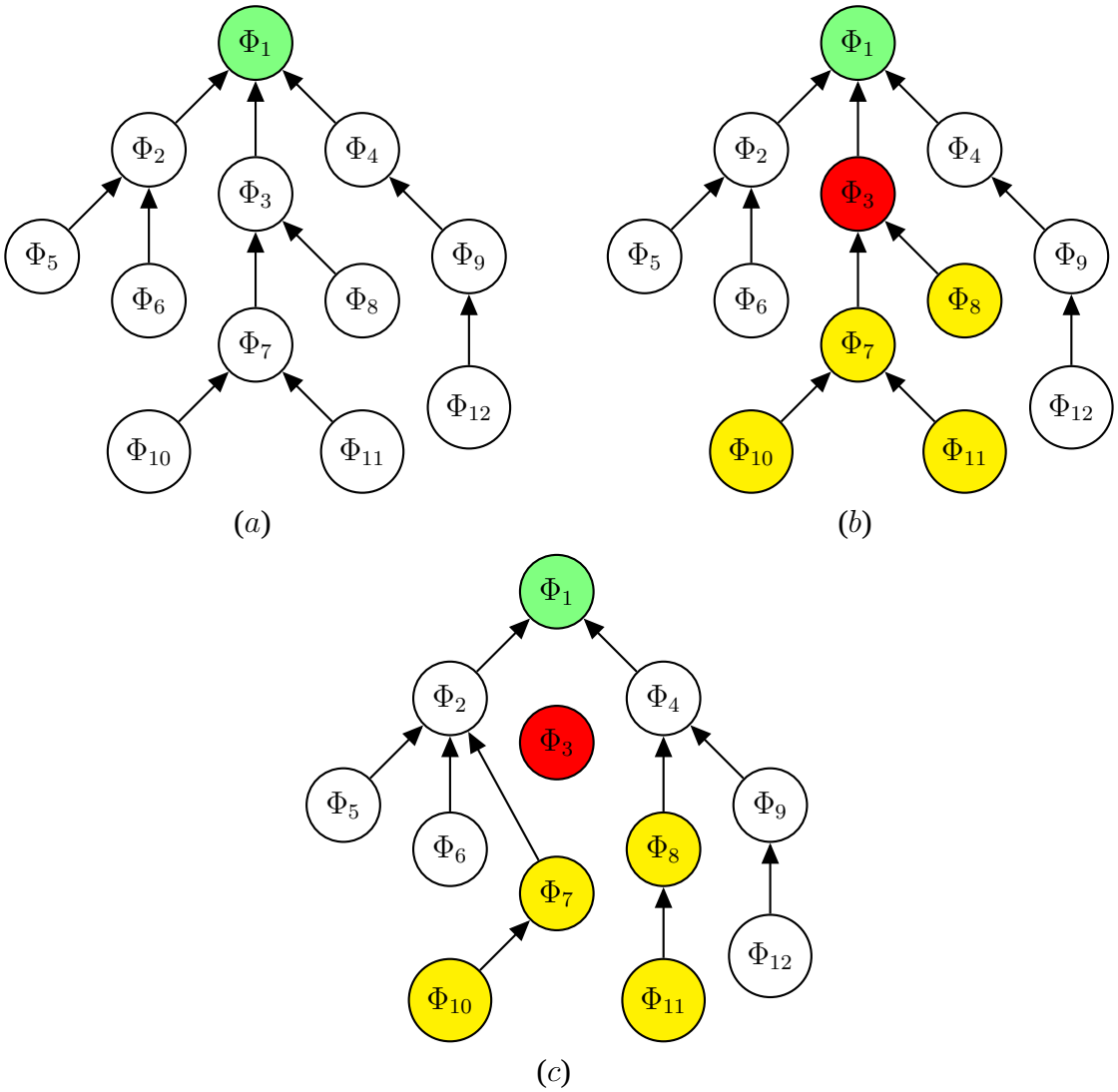


Figure 5.11: (a) Single-goal optimal motion policy tree with the goal motion policy Φ_1; (b) The motion policy Φ_3 is invalidated by a new obstacle, and hence the motion policies $\Phi_7, \Phi_8, \Phi_{10}$ and Φ_{11} need to be updated; and (c) Updated optimal motion policy tree. (Appeared in [79])

The simplest approach to account for the new obstacles is to regenerate the entire optimal motion policy tree while ignoring the invalid motion policy nodes as shown in Algorithm 5.2. Unlike Algorithm 5.1, Algorithm 5.2 optimizes only the valid motion policy nodes given by the *Valid*() function (*lines* $17 - 23$). In this approach, every time there is a change in the validity of motion policy nodes, the optimal motion policy tree has to be regenerated. For example, the motion policy nodes $\Phi_7, \Phi_8, \Phi_{10}, \Phi_{11}$ shown in Fig. 5.11(b) are the only nodes that had to be updated, whereas the other nodes did not require any updates. It is computationally inefficient to

regenerate the entire motion policy tree when only a subset of the motion policy nodes need to be updated. Therefore, this section presents an efficient dynamic replanning algorithm that updates only the motion policy nodes that need to be updated. This algorithm consists of the following three steps: (i) finding invalid motion policies, (ii) finding motion policy nodes to be updated, and (iii) updating the motion policy nodes.

Algorithm 5.2: Dijkstra's Algorithm with Invalid Nodes

```
   input : Gracefully Prepares Graph Ω
           Goal Motion Policy Node G
   output: Optimal Motion Policy Tree Γ
 1 begin
 2     Γ ← ∅
 3     foreach i ∈ Node(Ω) do
 4         Cost2Goal(i) ← ∞
 5         Next2Goal(i) ← ∅
 6         Γ ← Γ ∪ (i, Cost2Goal(i), Next2Goal(i))
 7     end
 8     Cost2Goal(G) ← 0
 9     Q ← Γ
10     while Q ≠ ∅ do
11         j ← MinCostNode(Q)
12         if Cost2Goal(j) = ∞ then
13             break
14         end
15         Q ← Q − {j}
16         foreach k ∈ Parent(j) do
17             if Valid(k) then
18                 c ← Cost2Goal(j) + EdgeCost(k, j)
19                 if c < Cost2Goal(k) then
20                     Cost2Goal(k) ← c
21                     Next2Goal(k) ← j
22                 end
23             end
24         end
25     end
26 end
```

5.3.4.1 Finding Invalid Motion Policies

The motion planner uses occupancy grids [18] to find the invalid cells in a finely discretized 2D map of the environment. The mapping from an occupancy cell to a motion policy in the motion policy library is obtained during the automatic motion policy instantiation procedure described in Sec. 5.3.1. Each occupancy cell has a list of motion policies whose domains cover it, and this list is updated for every valid motion policy instantiation. Therefore given a list of occupied cells, a simple look-up operation provides the list of invalid motion policy nodes I.

Algorithm 5.3: Find Nodes to be Updated

input : Base Optimal Motion Policy Tree Γ_0
Goal Motion Policy Node G
Set of Invalid Nodes I
output: Set of Nodes to be Updated U

```
begin
    U ← ∅
    foreach i ∈ I do
        U ← U ∪ i
    end
    Q ← G
    while Q ≠ ∅ do
        i ← Pop(Q)
        foreach j ∈ TreeParent(Γ_0, i) do
            if i ∈ U then
                U ← U ∪ j
            end
            Q ← Q ∪ j
        end
    end
end
```

5.3.4.2 Finding Motion Policy Nodes to be Updated

Figure 5.11(*b*) shows that the motion policy nodes that need to be updated belong to the subtree with the invalid motion policy node as its head. Any valid motion policy that reaches the goal motion policy via an invalid motion policy must be updated. This subtree of motion policy nodes can be found by traversing down the optimal motion policy tree of the base case, $i.e.$, the map

with all known static obstacles and no new obstacles. The algorithm used to find the motion policy nodes to be updated is shown in Algorithm 5.3. The base optimal motion policy tree Γ_0 is the best possible motion policy tree that can be generated, and is generated using Algorithm 5.1. Each motion policy node i has a list of motion policy nodes that point to it (parents) in the base optimal motion policy tree Γ_0 given by *TreeParent*(Γ_0, i). All invalid motion policy nodes are added to the list of nodes to be updated U (*lines* $3-5$), and the search for nodes to be updated begins with the goal motion policy node G and traverses down the entire base optimal motion policy tree Γ_0 in a breadth-first search manner (*lines* $6-15$). If a motion policy node i is marked to be updated, then all its parents in the base optimal tree Γ_0 given by *TreeParent*(Γ_0, i) are added to the list of nodes to be updated U (*lines* $10-12$).

5.3.4.3 Update the Motion Policy Nodes

The procedure to update the optimal motion policy tree Γ is presented in Algorithm 5.4. Every time the optimal motion policy tree Γ is to be updated, it is reset to the base optimal motion policy tree Γ_0, and the *Cost2Goal* values and *Next2Goal* pointers for all motion policy nodes in the list of nodes to be updated U are reset (*lines* $3-6$). For each valid motion policy node i in the list U given by *Valid*(i), its *Next2Goal* pointer is initialized to its valid child motion policy node with the minimum *Cost2Goal* (*lines* $9-17$). The children of each motion policy node are obtained from the gracefully prepares graph Ω. The invalid motion policy nodes are removed from the list U, while the valid motion policy nodes with the initialized *Cost2Goal* values and *Next2Goal* pointers are added to the list of unoptimized nodes Q. The procedure to optimize the *Cost2Goal* values for just these motion policy nodes (*lines* $25-40$) is same as that in Algorithm 5.1 (*lines* $10-23$) and in Algorithm 5.2 (*lines* $10-25$).

The dynamic replanning algorithm presented above is used only when the validity of a motion policy in the motion policy library changes. When such a change is detected, the current set of invalid motion policy nodes I is determined, and Algorithm 5.3 is used to find the motion policy nodes in the base optimal motion policy tree Γ_0 that need to be updated. The optimal motion policy tree Γ is then reset to Γ_0 and updated using Algorithm 5.4. Since all the motion policy nodes are updated whenever a change in validity of the motion policy nodes is detected, this dynamic replanning algorithm ensures that the resulting optimal control vector field for the bounded position state space covered by the motion policy library is always valid.

Algorithm 5.4: Update Optimal Motion Policy Tree

input : Base Optimal Motion Policy Tree Γ_0
Goal Motion Policy Node G
Set of Nodes to be Updated U
output: Updated Optimal Motion Policy Tree Γ

```
1  begin
2      Γ ← Γ_0
3      foreach i ∈ U do
4          Cost2Goal(i) ← ∞
5          Next2Goal(i) ← ∅
6      end
7      foreach i ∈ U do
8          if Valid(i) then
9              foreach j ∈ Child(i) do
10                 if Valid(j) then
11                     c ← Cost2Goal(j) + EdgeCost(i, j)
12                     if c < Cost2Goal(i) then
13                         Cost2Goal(i) ← c
14                         Next2Goal(i) ← j
15                     end
16                 end
17             end
18         else
19             U ← U − {i}
20         end
21     end
22     Q ← U
23     while Q ≠ ∅ do
24         j ← MinCostNode(Q)
25         if Cost2Goal(j) = ∞ then
26             break
27         end
28         Q ← Q − {j}
29         foreach k ∈ Parent(j) do
30             if Valid(k) then
31                 c ← Cost2Goal(j) + EdgeCost(k, j)
32                 if c < Cost2Goal(k) then
33                     Cost2Goal(k) ← c
34                     Next2Goal(k) ← j
35                 end
36             end
37         end
38     end
39 end
```

5.4 Experimental Results with The Ballbot

This section presents experimental results of the ballbot without arms successfully achieving different navigation tasks in a graceful manner using the integrated motion planning and control framework presented in Sec. 5.3.

5.4.1 Experimental Setup

The ballbot localizes itself on a 2D map of the environment using a particle filter based localization algorithm developed by Biswas *et al.* [6]. The encoders on the ball motors of the inverse mouse-ball drive [41] provide the odometry data for the prediction step, while a Hokuyo URG-04LX laser range finder provides the laser scan readings for the correction step.

The laser range finder is also used to detect obstacles in the environment, and its scan readings are used to update an occupancy grid map [18] with a grid spacing of 0.1 m. The laser range finder is mounted on the front of the robot at a height of 0.78 m from the floor, and has a 180° field of view. It has a linear resolution of 1 mm and an angular resolution of 0.36°. Since the laser range finder has only a 180° field of view, it cannot detect obstacles behind the robot. However, the ballbot can remember a previously encountered obstacle using its occupancy grid map. The occupancy grid map is primarily used to detect new static and dynamic obstacles, whereas permanent static obstacles are included in the map.

5.4.2 Motion Policy Library

In order to illustrate the integrated motion planning and control framework's capability to handle different motion policy palettes, this section presents experimental results that used two motion policy palettes, one with 39 unique motion policies and another with 43 unique motion policies. They used different motion primitive sets with the same distance parameter d = 0.5 m like the ones shown in Fig. 5.2. The palette with 43 motion policies had more non-stationary trim conditions than the palette with 39 motion policies. The problem of determining the number of motion policies required to optimally achieve a navigation task is an open research question, and is not addressed in this thesis. However, this problem must be explored as one of the future directions of research.

For all the results presented in this section, the motion policies were automatically instantiated in a 3.5 m $\times$ 3.5 m obstacle-free area of a map of our lab as described in Sec. 5.3.1.

The motion policy palette with 39 motion policies produces a motion policy library of 4521 instantiated motion policies, while the motion policy palette with 43 motion policies produces a motion policy library of 4569 instantiated motion policies. For both the motion policy palettes, the total time taken to generate the motion policy library and the gracefully prepares graph is approximately 2.5 s on the dual core computer on-board the robot. Similarly, the same computer takes about 0.05 s to generate the optimal motion policy tree for both the motion policy palettes. Therefore, the optimal motion policy tree can be updated as well as regenerated in real-time. All the results presented in this section use fastest time as the optimality criterion.

5.4.3 Point-Point Motion

The point-point navigation task can be formulated as a motion between trim motion policies with constant position reference trajectories as motion primitives. The goal motion policy in the motion policy library is given by the trim motion policy whose goal position state is the closest to the desired goal position by the Euclidean distance metric. Figure 5.12 shows the ballbot successfully reaching a single goal position state of (2 m, 2 m, 0 m/s, 0 m/s) from four different starting configurations in the presence of two static obstacles. The experimental position space motions shown in Fig. 5.12 were obtained from the localization algorithm [6].

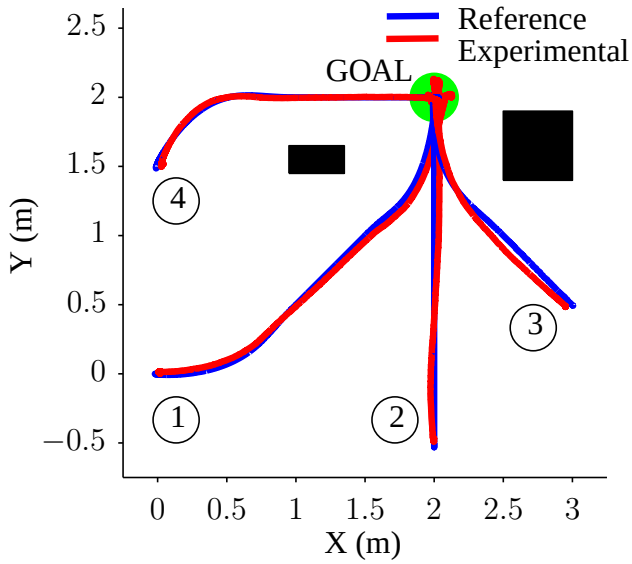


Figure 5.12: Point-Point motion with two obstacles, shown in black (Appeared in [78, 79]).

This navigation task used the motion policy palette with 39 motion policies to generate the

motion policy library. A single-goal time-optimal motion policy tree was generated online using Algorithm 5.2 that takes into account the known static obstacles in the map. Each of the motions shown in Fig. 5.12 were obtained by tracking motions from the single-goal time-optimal motion policy tree, and were not regenerated for each run.

Figure 5.13 shows the composite frames from a video of the ballbot achieving the goal from the first starting position, while the body angle trajectories resulting from the fourth motion are shown in Fig. 5.14. The video of the ballbot achieving all four motions can be found in Video D.7.

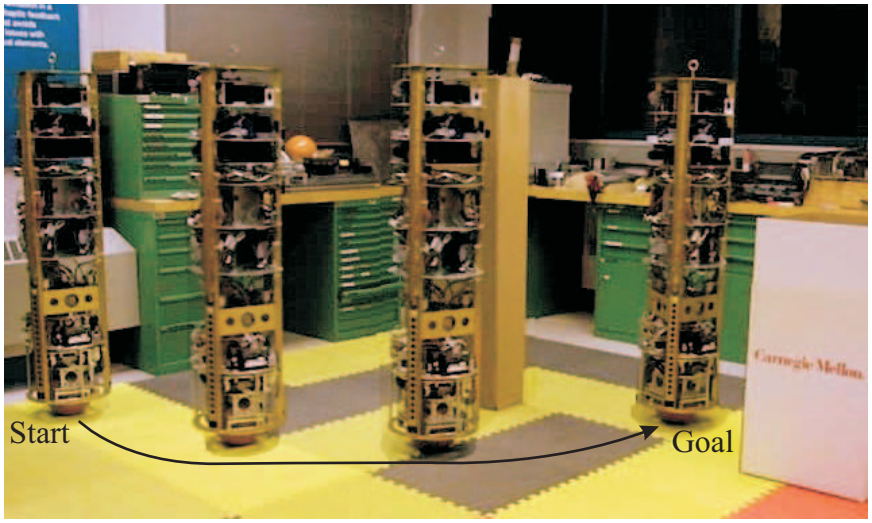


Figure 5.13: Composite frames from a video of the ballbot achieving the goal from start position no. 1 (Appeared in [79]).

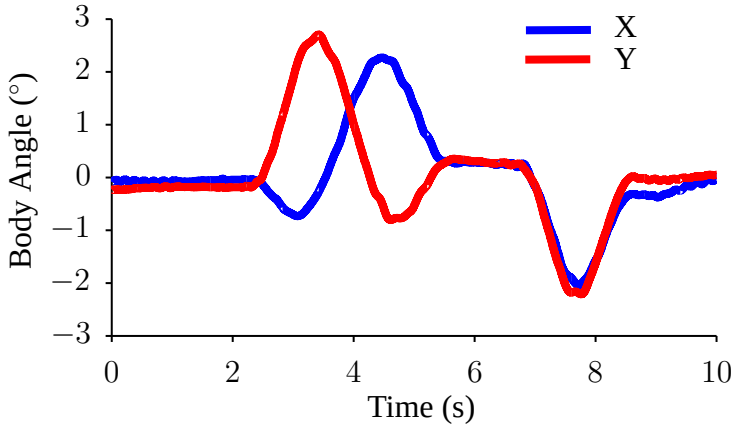


Figure 5.14: Point-Point motion: Body angle trajectories to achieve the goal from start position no. 4 (Appeared in [78]).

5.4.4 Disturbance Handling

Figure 5.15 presents the experimental results that illustrate the capability of the integrated motion planning and control framework to handle large disturbances. While the ballbot was executing a point-point motion, it was physically held and stopped from moving towards the goal. Then, the ballbot was physically moved to a different point on the map and let go.

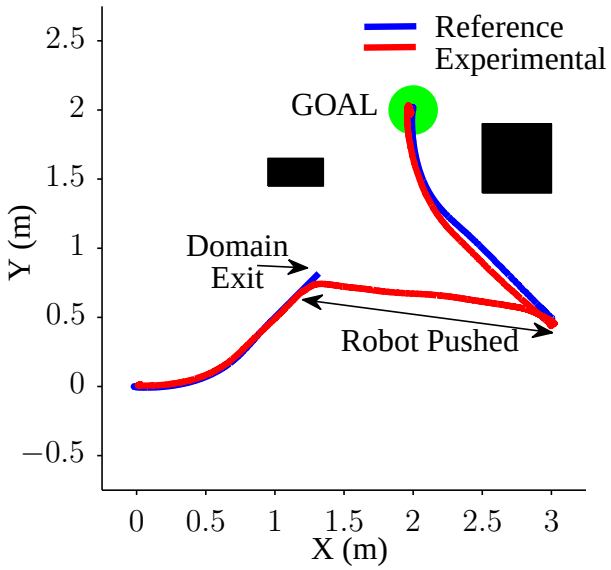


Figure 5.15: Disturbance Handling (Appeared in [78, 79]).

When the ballbot was physically stopped from moving towards the goal, its position state exited the domain of the motion policy it was executing, and the hybrid control architecture detected this domain exit. It then, searched the motion policy library to find motion policies whose start domain contained this exiting position state. Since a continued disturbance was applied to the robot, its position state continued to exit the domains that were found until the ballbot was set free to move on its own. After it was set free, the ballbot successfully reached the goal as shown in Fig. 5.15. Here again, the single-goal time-optimal motion policy tree was not regenerated. The composite frames from a video of the ballbot achieving this motion are shown in Fig. 5.16, and the video can be found in Video D.7.

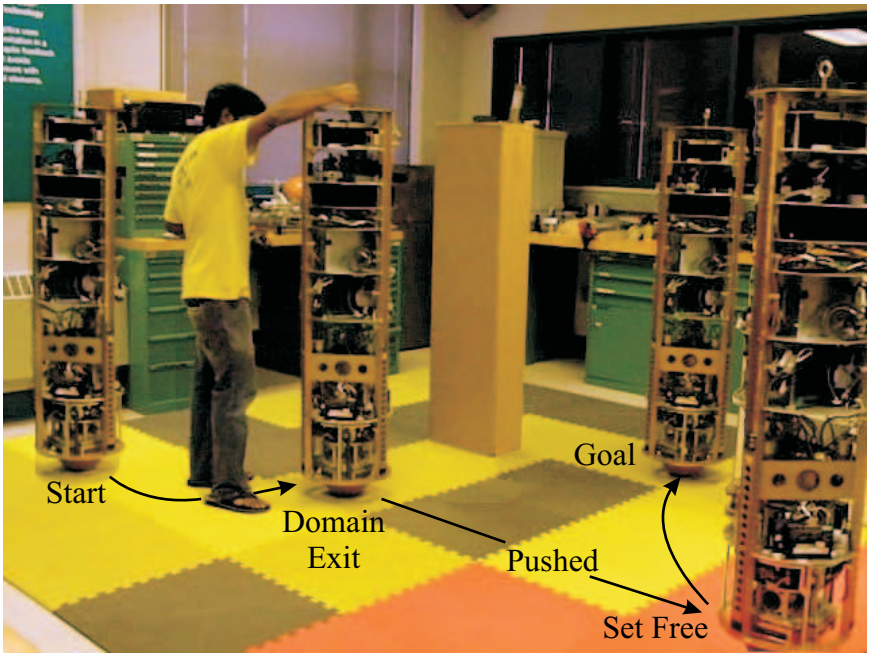


Figure 5.16: Composite frames from a video of the ballbot reaching the goal while handling a disturbance (Appeared in [79]).

5.4.5 Surveillance

The navigation task of surveillance can be formulated as a motion between trim motion policies that have constant velocity reference trajectories as motion primitives. The task is specified as a sequence of moving goal position states that are repeated in a cyclic fashion. Unlike point-point motion, the surveillance motion has multiple cyclic goals. A new goal motion policy is found every time the current goal motion policy is reached, and hence the time-optimal motion policy tree has to be regenerated. The motion planner takes only 0.05 s to regenerate the motion policy library used here, and all of its motion policies have time durations greater than or equal to 1 s. Hence, the optimal motion policy tree can be regenerated in real-time while executing the last motion policy to the current goal. This process repeats until the user quits the surveillance task.

Figure 5.17 shows the ballbot successfully performing a surveillance task with four goal configurations using the motion policy palette with 39 motion policies. In this run, the surveillance task was quit after five successful loops. The composite frames from a video of the ballbot achieving this task is shown in Fig. 5.18, and the video can be found in Video D.7. The body angle trajectories for one complete surveillance loop are shown in Fig. 5.19.

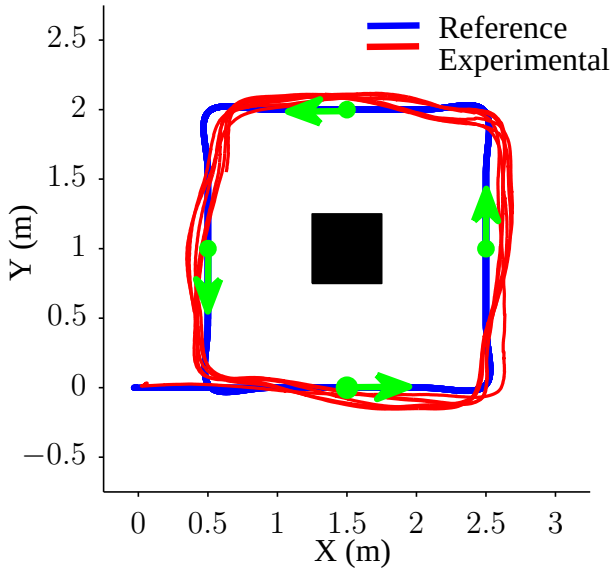


Figure 5.17: Surveillance motion with four goal configurations, shown in green and one obstacle, shown in black (Appeared in [78, 79]).

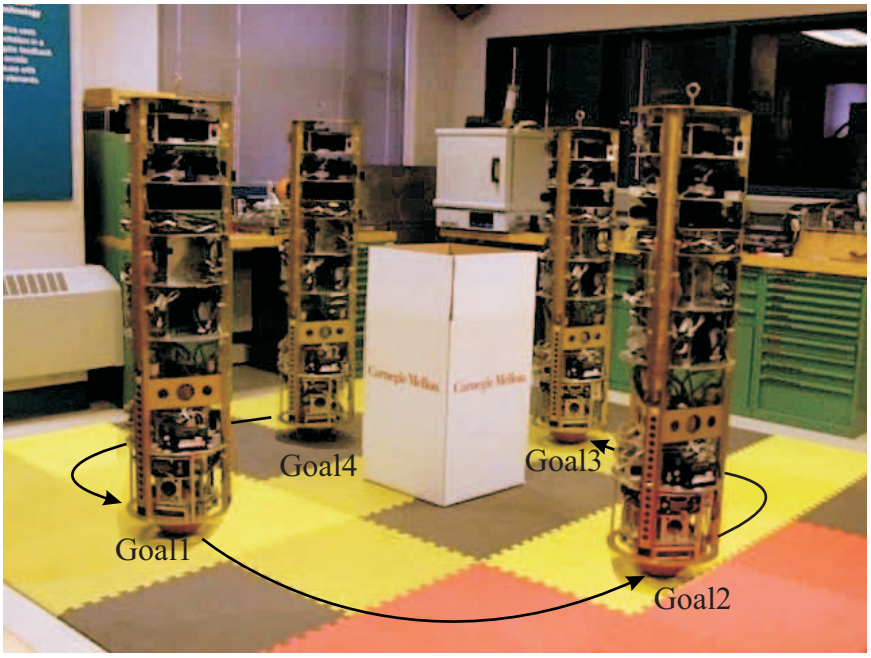


Figure 5.18: Composite frames from a video of the ballbot achieving the surveillance motion with four goal configurations (Appeared in [79]).

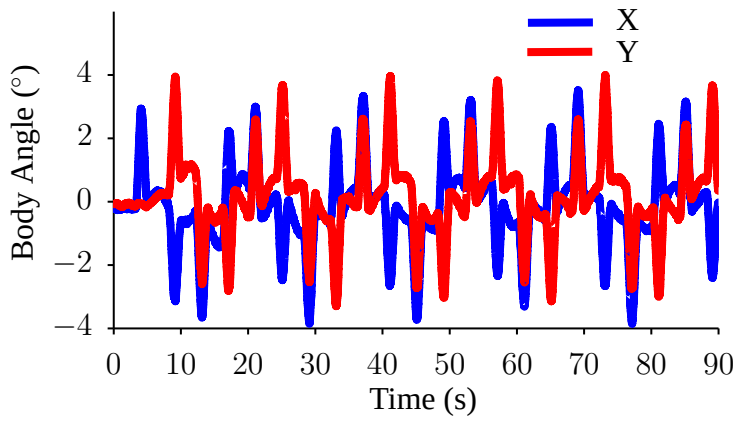


Figure 5.19: Body angle trajectories for the surveillance motion with four goal configurations (Appeared in [78, 79]).

Two successful loops of another surveillance task with ten goal configurations are shown in Fig. 5.20. This surveillance task was achieved using the motion policy palette with 43 motion policies, and a video of it can be found in Video D.8. The optimal motion policy tree was successfully regenerated real-time on the robot for every single goal configuration. The resulting body angle trajectories for one complete surveillance cycle are shown in Fig. 5.21.

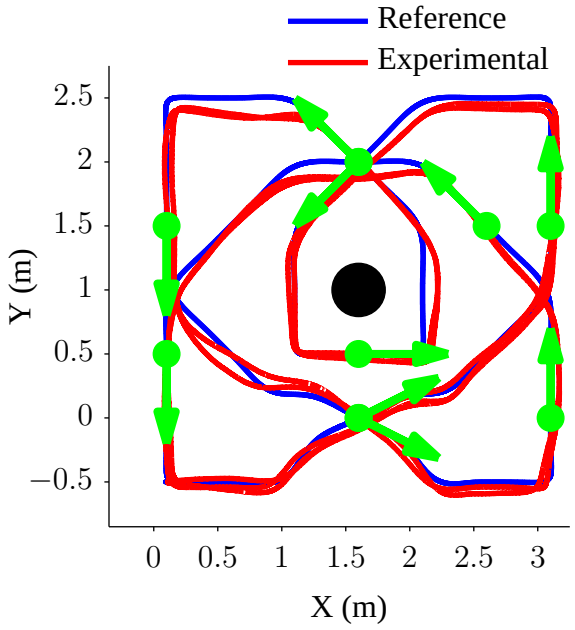


Figure 5.20: Surveillance motion with ten goal configurations, shown in green and one obstacle, shown in black (Appeared in [79]).

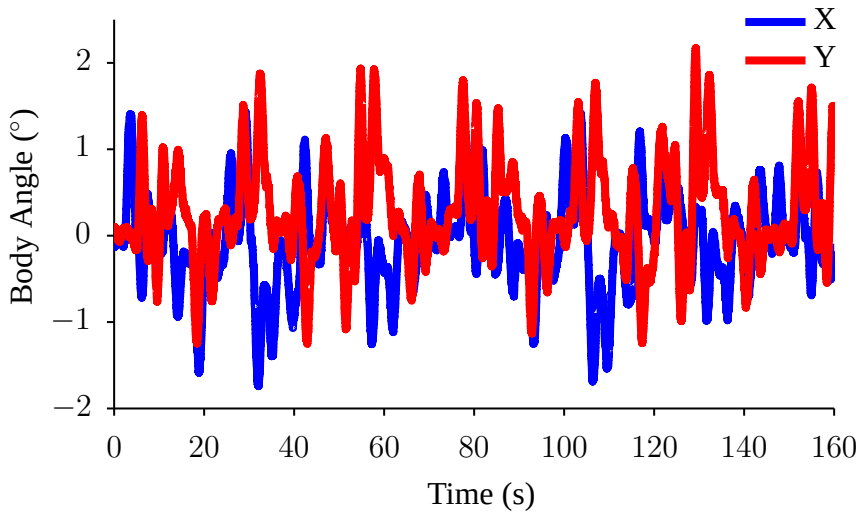


Figure 5.21: Body angle trajectories for the surveillance motion with ten goal configurations (Appeared in [79]).

5.4.6 Dynamic Replanning

Figure 5.22 shows the ballbot using the dynamic replanning algorithm presented in Sec. 5.3.4 to avoid new static and dynamic obstacles, and successfully reach the goal. The composite frames from a video of the ballbot performing this motion is shown in Fig. 5.23, and the video can be found in Video D.8. The base optimal reference motion to the goal without any new static or dynamic obstacles is shown in Fig. 5.24(a). The optimal reference motion to the goal with the new static obstacle and the dynamic obstacle in its first position is shown in Fig. 5.24(b). The optimal reference motion to the goal with the dynamic obstacle moving to its second location is shown in Fig. 5.24(c). The final optimal reference motion to the goal is shown in Fig. 5.24(d), and the ballbot's attempt at tracking this reference motion is shown in Fig. 5.22. It is important to note that the optimal motions achieved using this approach are limited by the motion policies in the motion policy library. For example, one can see that the optimal reference motion shown in Fig. 5.24(b) is not optimal in a true sense but is optimal w.r.t. the motion policies available in the motion policy library.

The motion policy palette with 43 motion policies was used for this run. The dual-core computer on-board the ballbot takes 0.01 s to 0.05 s to update the optimal motion policy tree depending on the number of motion policy nodes to be updated. Each motion policy is of time

duration greater than or equal to 1 s, and hence the optimal motion policy tree can be updated fast enough to account for the dynamic obstacles, which are detected by the laser range finder at 10 Hz, $i.e.$, every 0.1 s.

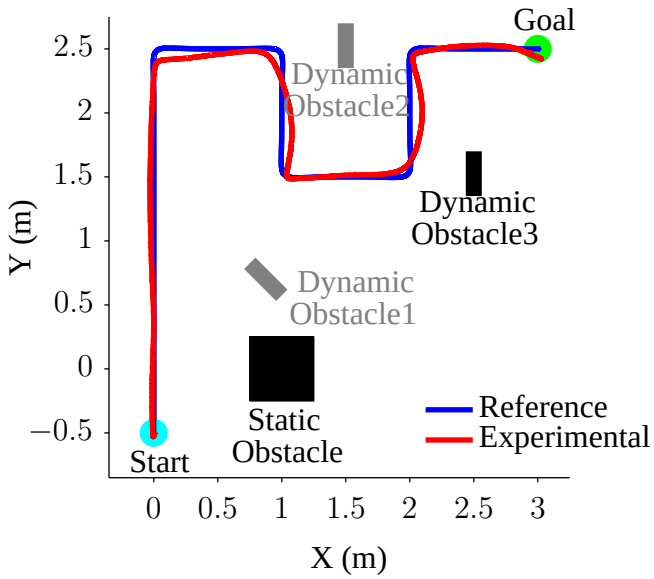


Figure 5.22: Dynamic replanning to reach the goal (Appeared in [79]).

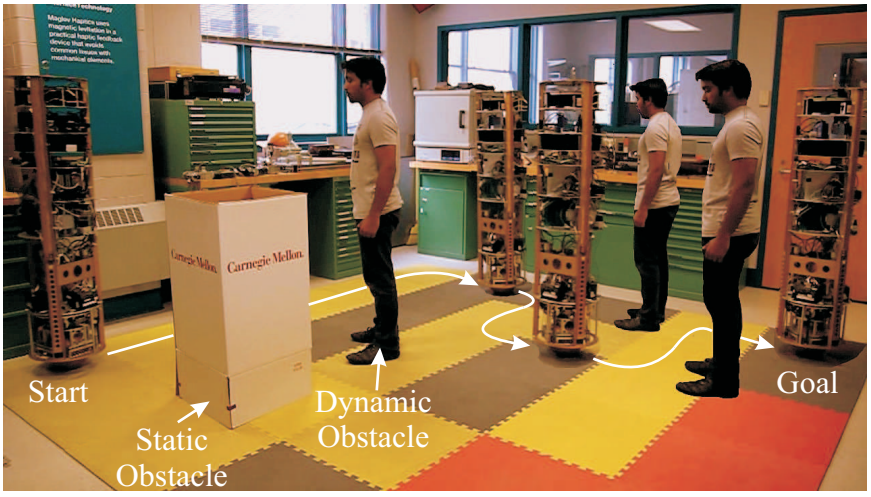


Figure 5.23: Composite frames from a video of the ballbot dynamically replanning to avoid static and dynamic obstacles (Appeared in [79]).

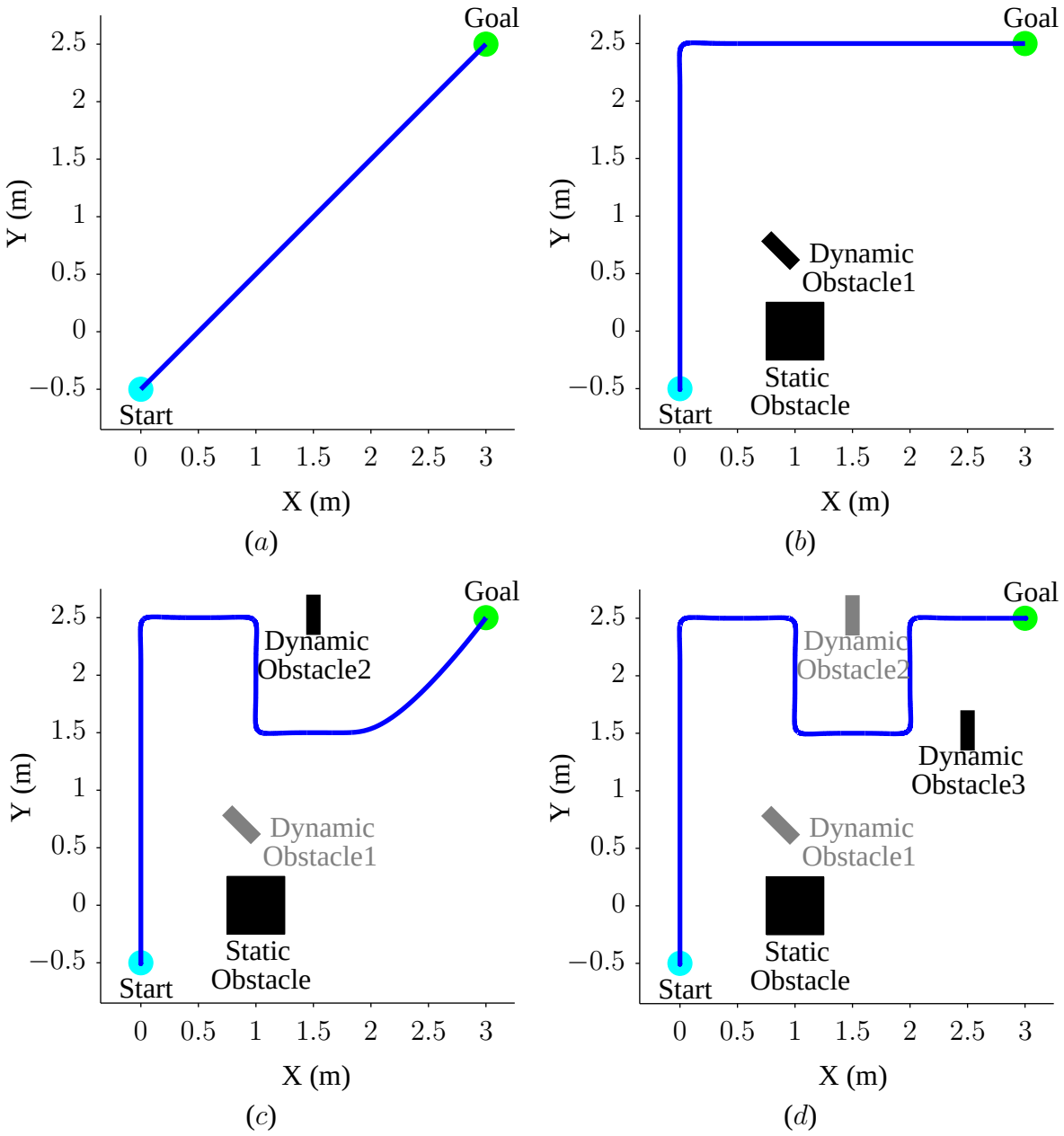


Figure 5.24: (a) The base optimal reference motion to the goal; (b) The optimal reference motion with the static obstacle, and the dynamic obstacle at its first location; (c) The optimal reference motion when the dynamic obstacle has moved to its second location; (d) The optimal motion when the dynamic obstacle has moved to its final location. (Appeared in [79])

5.5 Summary

This chapter presented an integrated motion planning and control framework that enables balancing mobile robots like the ballbot to gracefully navigate human environments. The *gracefully prepares relationship* was introduced as a restrictive definition on the prepares relationship. Unlike other sequential composition based approaches [10, 11, 77] that do not guarantee graceful switching between control policies, the integrated motion planning and control framework presented in this chapter uses the gracefully prepares relationship to ensure graceful switching between them. This chapter presented an offline procedure to design a palette of gracefully composable control policies called *motion policies* that result in collision-free graceful motions in small domains of the position space. It also presented an online automatic instantiation procedure that deploys these motion policies to fill a map of the environment, and generates a library of motion policies. A *gracefully prepares graph* that represented all possible graceful motions that the robot can achieve in a map was generated online, and navigation tasks were formulated as graph-search problems. This chapter used Dijkstra's algorithm to generate a single-goal optimal motion policy tree that represents an optimal control vector field for achieving the goal. However, the optimality is limited to the motion policies available in the motion policy library. This chapter also presented a dynamic replanning algorithm that replans the single-goal optimal motion policy tree in the presence of dynamic obstacles.

The most important contribution of this work is the experimental verification of the integrated motion planning and control framework on the ballbot. This chapter presented experimental results that demonstrate the ballbot successfully achieving a variety of different point-point and surveillance navigation tasks. This chapter also presented experimental results that demonstrate the capability of the framework to handle disturbances, and also actively replan in the presence of dynamic obstacles and still successfully achieve desired navigation tasks.

Chapter 6

Conclusions and Future Work

Balancing mobile robots can be tall and skinny with low centers of gravity and small footprints, unlike their statically stable counterparts. They can be tall enough for eye-level interaction and narrow enough to navigate cluttered human environments. They can actively resist tipping over by compensating for the shift in their centers of gravity. Their dynamic stability enables them to be physically interactive, and also enables them to achieve fast and graceful motions. All these characteristics make balancing mobile robots ideal candidates for personal robots operating and interacting in human environments.

The work presented in this thesis enables balancing mobile robots like the ballbot to navigate human environments with speed and grace comparable to that of humans. The approach taken to achieve this is two-fold: (i) the natural dynamics of balancing mobile robots are exploited to achieve fast and dynamic motions; and (ii) the motion planning and control are integrated to ensure that balancing mobile robots gracefully achieve desired navigation tasks while handling disturbances and dynamic obstacles.

Fast and dynamic motions in the position space are achieved by planning for motions in the shape space because the shape dynamics dominate the system dynamics. A palette of sequentially composable controllers called *motion policies* are designed such that their valid compositions produce graceful robot motions. They are designed such that they have knowledge of their reference motions in the state space and their collision-free domains in the position state space. Desired navigation tasks are achieved by planning in the space of these gracefully composable motion policies.

This chapter highlights the contributions of the work presented in this thesis, and also discusses several future directions that can be explored to extend and better the shape trajectory

planner, and the integrated motion planning and control framework presented in this thesis.

6.1 Contributions

The contributions of the work presented in this thesis are listed below.

(i) One of the first contributions of this work was the design of the balancing controller and other controllers that enabled the ballbot to balance and operate reliably [74, 76]. The balancing controller enabled the ballbot to be robust to disturbances like shoves, kicks and collisions with furniture and walls, and it also enabled the ballbot to be physically interactive. Chapter 3 presented successful experimental results on the ballbot that demonstrated its capabilities, and also demonstrated some interesting human-robot physical interaction behaviors.

(ii) Chapter 4 introduced a special class of underactuated systems called *shape-accelerated balancing systems* [72, 73] to which balancing mobile robots like the ballbot belong. These systems have a special property wherein non-zero shape configurations result in accelerations in the position space, and this special property can be exploited to enable balancing mobile robots like the ballbot to achieve fast and dynamic motions.

(iii) Chapter 4 presented a novel shape trajectory planner [72] that plans shape trajectories for shape-accelerated balancing systems, which when tracked will result in optimal tracking of desired acceleration trajectories in position space. The trajectory optimization algorithm plans shape trajectories that are linearly proportional to desired acceleration trajectories in the position space, and the proportionality gains are determined such that the acceleration trajectories that result from tracking these planned shape trajectories will optimally track the desired acceleration trajectories in position space. It was demonstrated that the shape trajectory planner can generate feasible trajectories for shape-accelerated balancing systems at significantly faster speeds (25 to 70 times) than other trajectory optimization algorithms using direct-collocation methods [30, 35, 104, 130, 131]. Such fast computation times were possible because the shape trajectory planner optimized in a significantly lower dimensional parameter space, and also used only the dynamic constraint equations of the system for numerical integration.

(iv) Chapter 4 also introduced the notion of a *shape set* [73], a group of shape variables that are capable of independently affecting the dynamics of all position variables. It was demon-

strated that the shape trajectory planner can successfully plan motions in the shape space for systems with equal number of shape and position variables, $i.e.$, systems with just one shape set. The shape trajectory planner was extended to successfully plan in high-dimensional shape space with the use of user-defined weight matrices, where the user picks the relative contribution between different shape sets [73, 81]. It was also extended to include cases where a subset of the system's shape sets were artificially constrained to desired trajectories [73].

(v) Chapter 4 presented successful experimental results that validated the shape trajectory planner and its associated control architecture on the ballbot, both with and without arms [73, 81]. The ballbot was able to successfully achieve a variety of different fast and dynamic motions using its body lean motions. It also successfully achieved desired motions in the position space using pure arm motions, and combinations of body lean and arm motions. Successful experimental results that demonstrate the capability of the ballbot to achieve desired motions in the position space using only the body lean motions, while its arms were constrained to desired trajectories were also presented.

(vi) Chapter 5 introduced the *gracefully prepares relationship* [78, 79] between two control policies, as a restrictive definition on the prepares relationship [10, 11, 77]. The gracefully prepares relationship guaranteed that the closed-loop motion of the system resulting from a valid sequential composition of control policies was graceful. This was achieved by ensuring graceful composition of their reference motions, feedback control laws, and domains.

(vii) Chapter 5 presented an integrated motion planning and control framework that enabled balancing mobile robots like the ballbot to gracefully achieve desired navigation tasks while handling disturbances and dynamic obstacles [78, 79]. It presented an offline design procedure for controllers called *motion policies* that produce graceful, collision-free motions in small domains of the position space for shape-accelerated balancing systems like the ballbot. A palette of motion policies was designed such that its motion policies gracefully prepare each other. An automatic instantiation procedure that uniformly distributes motion policies from a palette to fill a map of the environment was presented. The domain of each motion policy in the motion policy library was verified to be collision-free. Chapter 5 also introduced a directed graph called the *gracefully prepares graph* that represents all graceful, collision-free motions that the robot can achieve on the map using the motion policy library. A motion planner that plans in the space of these collision-free, gracefully

composable motion policies was presented. The navigation tasks were formulated as motions between motion policies, and Dijkstra's algorithm was used to generate a single-goal optimal motion policy tree. The optimal motion policy tree represented optimal graceful motions to the goal from the start domain of every motion policy in the motion policy library. A dynamic replanning algorithm that quickly replans in the presence of dynamic obstacles was also presented.

($viii$) Chapter 5 presented successful experimental results that validated the integrated motion planning and control framework on the ballbot without arms. The ballbot was able to successfully achieve a number of point-point and surveillance navigation tasks in the presence of static obstacles. Successful experimental results that demonstrate the framework's capability to handle disturbances were presented. The ballbot was also able to successfully achieve point-point navigation tasks in the presence of dynamic obstacles, where it had to dynamically replan the optimal motion policy tree in real-time while successfully avoiding the dynamic obstacles.

6.2 Future Work

This section presents brief descriptions of different improvements and extensions that can be applied to the work presented in this thesis.

6.2.1 Shape Space Planning with Manipulation

The shape trajectory planner presented in Chapter 4 can be extended to plan for a combination of manipulation and navigation tasks. A variant of the shape trajectory planner that achieves desired navigation tasks for the ballbot using only body lean motions while the arms were restricted to additional constraint trajectories was presented in Sec. 4.2.4, and experimentally verified in Sec. 4.3.4. However, no manipulation tasks were performed. For balancing mobile robots like the ballbot, navigation and manipulation tasks are tightly coupled, and some of the challenges that arise due to this coupling are described below.

One of the challenges is that the weight of the object that is manipulated plays a significant role in the robot's balance and in its navigation. A state estimator that actively estimates the position of the net center of gravity of the robot and the object is essential for successfully performing such tasks. The shape trajectory planner will have to take into account the manipulation

trajectories, and also the dynamics associated with the object's motions.

Another challenge is that the robot's body lean motions will affect its manipulation trajectories as the robot will lean with its arms attached to its body. Let's take an example case where the balancing robot has to pick up an object on a table while staying in place. While the arms move forward to pick up the object, the body has to lean backward in order to stay in place. But, the arms which are attached to the body will lean backward with it and hence, its end-effector may not reach its correct location in the world. This will happen if the manipulation trajectory planner ignores the body motions required to stay balanced while planning manipulation trajectories. A trajectory optimization algorithm that combines the shape trajectory planner and the manipulation trajectory planner can be developed such that both the navigation and manipulation tasks are achieved. One approach towards developing such a trajectory optimization algorithm is described below. At every iteration, the optimization algorithm determines a pseudo-goal for the manipulation planner, which compensates for the shift in the position of the end-effector as a result of body lean motions. Given a pseudo-goal for the end-effector, the manipulation planner uses the inverse kinematics of a manipulator with a stationary base to plan trajectories for the arm angles. The shape trajectory planner considers the manipulation trajectories as constraint trajectories for the arm angles, and plans body lean trajectories that achieve the desired navigation task as described in Sec. 4.2.4. Now, the resulting position of the end-effector in the world is determined and compared with its ideal goal position given by the manipulation task. The optimization algorithm uses this difference to update the pseudo-goal for the manipulation planner in its next iteration.

6.2.2 Navigating Large Maps

The integrated motion planning and control framework presented in Chapter 5 used the Dijkstra's algorithm for motion planning in the space of gracefully composable motion policies. But the Dijkstra's algorithm may not be fast enough on bigger graphs that cover larger areas. In order to handle large maps, two different approaches can be explored. One approach involves the use of heuristic based graph search algorithms like D^* [121] and D^* *Lite* [53]. The design of admissible heuristics in the space of motion policies is a challenging problem and must be explored.

The other approach involves dividing the large map into smaller regions, and piecing together locally optimal motions to achieve the global goal. However, this approach of using regions still requires a heuristic to determine the optimal path in the space of regions. In order for such an approach to be successful, it must guarantee that the robot will neither be trapped in a region nor

be stuck in a cycle between regions. A simple version of this proposed approach of using regions was successfully attempted on the ballbot to navigate larger maps, like a long corridor (20 m $\times$ 2.4 m area), moving between rooms in our lab, and also a 9.5 m $\times$ 7.5 m rectangular area. The videos of the ballbot successfully navigating these larger maps can be found in Video D.9. However, the challenges of finding the best heuristic, and also handling the corner cases like region traps and region cycles were not addressed, and must be explored.

6.2.3 Design of Invariant Motion Policy Domains

One of the improvements that can be done to the integrated motion planning and control framework presented in Chapter 5 is in the design and verification of the motion policy domains. The current advancements in algebraic verification of controllers using sums-of-squares (SOS) tools [125, 128] can be used to design invariant domains for the motion policies. It is difficult to achieve 100% coverage in environments with narrow paths as the motion policy domain definitions presented in this work are fixed. But the invariant motion policy domain definitions will allow one to explore ways to scale-down these motion policy domains during instantiation so that the obstacles can be avoided, and the desired coverage can also be guaranteed.

6.2.4 Optimal Palette of Motion Policies

A procedure to design a palette of gracefully composable motion policies for shape-accelerated balancing systems like the ballbot was presented in Sec. 5.2. However, this section did not present any insight into choosing a motion policy palette for a given navigation task. Given a navigation task, the minimum number and type of motion policies required to achieve it, and the optimal set of motion policies that can achieve it are open research questions that need to be addressed. Moreover, metrics that can compare and evaluate two motion policy palettes for a navigation task must also be explored.

6.2.5 Integrated Motion Planning and Control for Graceful Manipulation

Chapter 5 presented an integrated motion planning and control framework that ensures graceful navigation of balancing mobile robots, and Section 5.4 presented experimental results of the ballbot without arms using this framework to successfully achieve desired navigation tasks. The integrated motion planning and control framework must be extended to include manipulation

tasks as well. Each reference position space motion can have a variety of different manipulation tasks associated with it, and hence a variety of different motion policies that produce the same position space motion but different arm motions can be generated. The definition of gracefully prepares relationship remains valid, but the motion policy domains can no longer be restricted to the 4D position state space. In order to guarantee both graceful navigation and graceful manipulation, the motion policy domains have to be defined for the arm configurations as well. In these cases, each trim condition will consist of both a navigation goal and a manipulation goal, and the motion planner has to plan in the space of these gracefully composable motion policies to achieve the desired navigation and manipulation tasks.

Appendix A

Dynamic model for the 3D ballbot with a pair of 2-DOF arms

The ballbot with arms is modeled as a rigid cylinder on top of a rigid sphere with a pair of massless arms having weights at their ends. The model makes the following assumptions: (i) there is no slip between the ball and the floor; and (ii) there is no yaw/spinning motion for either the ball or the body or the arms, $i.e.$, they have two degrees of freedom each. A planar version of the model along with the planar configurations is shown in Fig. A.1.

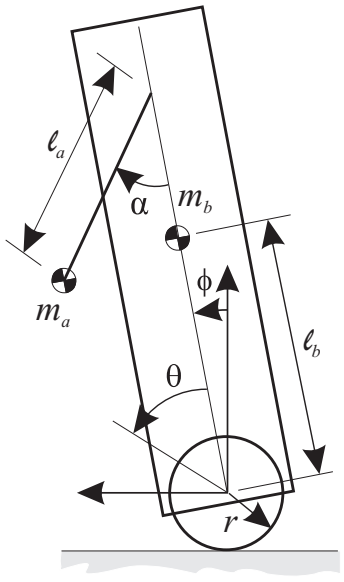


Figure A.1: Planar configurations shown in a planar model of ballbot with arm (Appeared in [73, 81]).

There are eight configuration variables for the 3D ballbot model with arms represented by $q = [\theta, \alpha_l, \alpha_r, \phi]$, where, $\theta = [\theta_x, \theta_y]^T$ are configurations of the ball, $\alpha_l = [\alpha_{lx}, \alpha_{ly}]^T$ are configurations of the left arm, $\alpha_r = [\alpha_{rx}, \alpha_{ry}]^T$ are configurations of the right arm, and $\phi = [\phi_x, \phi_y]^T$ are configurations of the body. The forced Euler-Lagrange equations of motion of the ballbot with arms can be written in matrix form as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}, \tag{A.1}$$

where, $M(q) \in \mathbb{R}^{8\times 8}$ is the mass/inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{8\times 8}$ is the matrix of Coriolis and centrifugal terms, $G(q) \in \mathbb{R}^{8\times 1}$ is the vector of gravitational forces and $\tau = [\tau_\theta, \tau_{\alpha_l}, \tau_{\alpha_r}]^T \in \mathbb{R}^{6\times 1}$ is the vector of generalized forces. The body configurations ϕ are unactuated, whereas the rest of the configurations are actuated.

The remainder of this chapter will present the system matrices and their elements as functions of the system parameters shown in Table A.1. The following symbols are used in their expressions: $\beta = I_w + \big(m_w + m_b + 2m_a\big)r^2$, $\gamma_1 = m_b\ell_b r$, $\gamma_2 = m_a\ell_a r$, $\gamma_3 = m_a d_a^z r$, $\eta_1 = m_b\ell_b^2$, $\eta_2 = m_a\ell_a^2$, $\eta_3 = m_a\ell_a d_a^z$, $\eta_4 = m_a\ell_a d_a^y$, $\eta_5 = m_a {d_a^z}^2$, $\eta_6 = m_a d_a^{y2}$, $\chi_1 = m_b g\ell_b$, $\chi_2 = m_a g\ell_a$, $\chi_3 = m_a g d_a^z$, $C_i = \cos(i)$, and $S_i = \sin(i)$.

Table A.1: System Parameters for The Ballbot with Arms

Parameter	**Symbol**	**Value**
Ball radius	r	0.106 m
Ball mass	m_w	2.44 kg
Ball moment of inertia	I_w	0.0174 kgm^2
Body mass	m_b	70.3 kg
Body CM along z-axis from ball center	ℓ_b	0.87 m
Body roll moment of inertia about CM	I_{xx}^b	12.59 kgm^2
Body pitch moment of inertia about CM	I_{yy}^b	12.48 kgm^2
Body yaw moment of inertia about CM	I_{zz}^b	0.66 kgm^2
Arm mass	m_a	1 kg
Arm length from joint to CM	ℓ_a	0.55 m
Arm joint distance along y-axis from ball center	d_a^y	0.18415 m
Arm joint distance along z-axis from ball center	d_a^z	1.3 m
Arm roll moment of inertia about CM	I_{xx}^a	0.0016 kgm^2
Arm pitch moment of inertia about CM	I_{yy}^a	0.0016 kgm^2
Arm yaw moment of inertia about CM	I_{zz}^a	0.0010 kgm^2

The mass/inertia matrix $M(q)$ in Eq. A.1 is given by:

$$M(q) = \begin{bmatrix} M_{\theta\theta} & M_{\theta\alpha^l}(q_s) & M_{\theta\alpha^r}(q_s) & M_{\theta\phi}(q_s) \\ M_{\alpha^l\theta}(q_s) & M_{\alpha^l\alpha^l}(q_s) & M_{\alpha^l\alpha^r}(q_s) & M_{\alpha^l\phi}(q_s) \\ M_{\alpha^r\theta}(q_s) & M_{\alpha^r\alpha^l}(q_s) & M_{\alpha^r\alpha^r}(q_s) & M_{\alpha^r\phi}(q_s) \\ M_{\phi\theta}(q_s) & M_{\phi\alpha^l}(q_s) & M_{\phi\alpha^r}(q_s) & M_{\phi\phi}(q_s) \end{bmatrix}, \tag{A.2}$$

where each $M_{ij} \in \mathbb{R}^{2\times 2}$ is given below.

$$M_{\theta\theta}(q) = \left[\begin{array}{c:c} \beta & 0 \\ \hdashline 0 & \beta \end{array}\right], \tag{A.3}$$

$$M_{\theta\alpha^l}(q) = \left[\begin{array}{c:c} \begin{array}{c}\gamma_2\big(C_{\phi_y}S_{\alpha_x^l}S_{\alpha_y^l} + S_{\phi_x}S_{\phi_y}C_{\alpha_x^l}\big) \\ +\gamma_2 C_{\phi_x}S_{\phi_y}S_{\alpha_x^l}C_{\alpha_y^l}\end{array} & -\gamma_2 C_{\alpha_x^l}\big(C_{\phi_y}C_{\alpha_y^l} - C_{\phi_x}S_{\phi_y}S_{\alpha_y^l}\big) \\ \hdashline \gamma_2\big(C_{\phi_x}C_{\alpha_x^l} - S_{\phi_x}S_{\alpha_x^l}C_{\alpha_y^l}\big) & -\gamma_2 S_{\phi_x}C_{\alpha_x^l}S_{\alpha_y^l} \end{array}\right], \tag{A.4}$$

$$M_{\theta\alpha^r}(q) = \left[\begin{array}{c:c} \begin{array}{c}\gamma_2\big(C_{\phi_y}S_{\alpha_x^r}S_{\alpha_y^r} + S_{\phi_x}S_{\phi_y}C_{\alpha_x^r}\big) \\ +\gamma_2 C_{\phi_x}S_{\phi_y}S_{\alpha_x^r}C_{\alpha_y^l}\end{array} & -\gamma_2 C_{\alpha_x^r}\big(C_{\phi_y}C_{\alpha_y^r} - C_{\phi_x}S_{\phi_y}S_{\alpha_y^r}\big) \\ \hdashline \gamma_2\big(C_{\phi_x}C_{\alpha_x^r} - S_{\phi_x}S_{\alpha_x^r}C_{\alpha_y^r}\big) & -\gamma_2 S_{\phi_x}C_{\alpha_x^r}S_{\alpha_y^r} \end{array}\right], \tag{A.5}$$

$$M_{\theta\phi}(q) = \left[\begin{array}{c:c} \begin{array}{c} -\big(\gamma_1+2\gamma_3\big)S_{\phi_x}S_{\phi_y} \\ +\gamma_2 C_{\phi_x}S_{\phi_y}\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big) \\ +\gamma_2 S_{\phi_x}S_{\phi_y}\big(C_{\alpha_x^l}C_{\alpha_y^l}+C_{\alpha_x^r}C_{\alpha_y^r}\big) \end{array} & \begin{array}{c} \beta+\big(\gamma_1+2\gamma_3\big)C_{\phi_x}C_{\phi_y} \\ +\gamma_2 S_{\phi_y}\big(C_{\alpha_x^l}S_{\alpha_y^l}+C_{\alpha_x^r}S_{\alpha_y^r}\big) \\ +\gamma_2 S_{\phi_x}C_{\phi_y}\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big) \\ -\gamma_2 C_{\phi_x}C_{\phi_y}\big(C_{\alpha_x^l}C_{\alpha_y^l}+C_{\alpha_x^r}C_{\alpha_y^r}\big) \end{array} \\ \hdashline \begin{array}{c} -\beta-\big(\gamma_1+2\gamma_3\big)C_{\phi_x} \\ +\gamma_2 C_{\phi_x}\big(C_{\alpha_x^l}C_{\alpha_y^l}+C_{\alpha_x^r}C_{\alpha_y^r}\big) \\ -\gamma_2 S_{\phi_x}\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big) \end{array} & 0 \end{array}\right], \quad \text{(A.6)}$$

$$M_{\alpha^l\theta}(q) = M_{\theta\alpha_l}^T(q), \quad \text{(A.7)}$$

$$M_{\alpha^l\alpha^l}(q) = \left[\begin{array}{c:c} \eta_2 + I_{xx}^a C_{\alpha_y^l}^2 + I_{zz}^a S_{\alpha_y^l}^2 & 0 \\ \hdashline 0 & \eta_2 C_{\alpha_x^l}^2 + I_{yy}^a \end{array}\right], \quad \text{(A.8)}$$

$$M_{\alpha^l\alpha^r}(q) = \left[\begin{array}{c:c} 0 & 0 \\ \hdashline 0 & 0 \end{array}\right], \quad \text{(A.9)}$$

$$M_{\alpha^l\phi}(q) = \left[\begin{array}{c:c} \begin{array}{c} \eta_2 C_{\alpha_y^l} - \eta_3 C_{\alpha_x^l} + \eta_4 S_{\alpha_x^l} \\ \gamma_2\big(S_{\phi_x}S_{\alpha_x^l}C_{\alpha_y^l} - C_{\phi_x}C_{\alpha_x^l}\big) \end{array} & \begin{array}{c} \big(\eta_3 C_{\phi_x} + \eta_4 S_{\phi_y} + \gamma_2 C_{\phi_y}\big)S_{\alpha_x^l}S_{\alpha_y^l} \\ +\gamma_2 S_{\phi_y}\big(S_{\phi_x}C_{\alpha_x^l} + C_{\phi_x}S_{\alpha_x^l}C_{\alpha_y^l}\big) \\ +\eta_2 S_{\phi_x}S_{\alpha_y^l} \end{array} \\ \hdashline \begin{array}{c} \big(\eta_4 + \eta_2 S_{\alpha_x^l} + \gamma_2 S_{\phi_x}\big)C_{\alpha_x^l}S_{\alpha_y^l} \end{array} & \begin{array}{c} -\big(\eta_3 C_{\phi_x} + \eta_4 S_{\phi_x}\big)C_{\alpha_x^l}C_{\alpha_y^l} \\ +\eta_2 C_{\alpha_x^l}\big(C_{\phi_x}C_{\alpha_x^l} - S_{\phi_x}S_{\alpha_x^l}C_{\alpha_y^l}\big) \\ -\gamma_2 C_{\alpha_x^l}\big(C_{\phi_y}C_{\alpha_y^l} - C_{\phi_x}S_{\phi_y}S_{\alpha_y^l}\big) \end{array} \end{array}\right], \quad \text{(A.10)}$$

$$M_{\alpha^r\theta}(q) = M^T_{\theta\alpha^r}(q) \tag{A.11}$$

$$M_{\alpha^r\alpha^l}(q) = M^T_{\alpha^l\alpha^r}(q) \tag{A.12}$$

$$M_{\alpha^r\alpha^r}(q) = \left[\begin{array}{c:c} \eta_2 + I^a_{xx}C^2_{\alpha^r_y} + I^a_{zz}S^2_{\alpha^r_y} & 0 \\ \hdashline 0 & \eta_2 C^2_{\alpha^r_x} + I^a_{yy} \end{array}\right], \tag{A.13}$$

$$M_{\alpha^r\phi}(q) = \left[\begin{array}{c:c} \begin{array}{c}\big(\eta_2 - \eta_4 S_{\alpha^r_x} + \gamma_2 S_{\phi_x} S_{\alpha^r_x}\big)C_{\alpha^r_y} \\ -\big(\eta_3 + \gamma_2 C_{\phi_x}\big)C_{\alpha^r_x}\end{array} & \begin{array}{c}\eta_2 S_{\phi_x} S_{\alpha^r_y} \\ +\big(\gamma_2 C_{\phi_y} + \eta_3 C_{\phi_x} - \eta_4 S_{\phi_x}\big)S_{\alpha^r_x} S_{\alpha^r_y} \\ +\gamma_2 S_{\phi_y}\big(S_{\phi_x} C_{\alpha^r_x} + C_{\phi_x} S_{\alpha^r_x} C_{\alpha^r_y}\big)\end{array} \\ \hdashline \begin{array}{c}\big(\gamma_2 S_{\phi_x} + \eta_2 S_{\alpha^r_x} - \eta_4\big)C_{\alpha^r_x} S_{\alpha^r_y}\end{array} & \begin{array}{c}\big(\eta_4 S_{\phi_x} - \eta_2 S_{\phi_x} S_{\alpha^r_x}\big)C_{\alpha^r_x} C_{\alpha^r_y} \\ +\big(\gamma_2 S_{\phi_y} S_{\alpha^r_y} - \eta_3 C_{\alpha^r_y}\big)C_{\phi_x} C_{\alpha^r_x} \\ +\big(\eta_2 C_{\phi_x} C_{\alpha^r_x} - \gamma_2 C_{\phi_y} C_{\alpha^r_y}\big)C_{\alpha^r_x}\end{array} \end{array}\right], \tag{A.14}$$

$$M_{\phi\theta}(q) = M^T_{\theta\phi}(q) \tag{A.15}$$

$$M_{\phi\alpha^l}(q) = M^T_{\alpha^l\phi}(q) \tag{A.16}$$

$$M_{\phi\alpha^r}(q) = M^T_{\alpha^r\phi}(q) \tag{A.17}$$

$$
M_{\phi\phi}(q) = \left[\begin{array}{c:c}
\begin{array}{c}
I^b_{zz} S^2_{\phi_y} + 2\eta_4\big(S_{\alpha^l_x} - S_{\alpha^r_x}\big) \\
+I^b_{xx} C^2_{\phi_y} + \big(2\gamma_1 + 4\gamma_3\big) C_{\phi_x} \\
-2\eta_3\big(C_{\alpha^l_x} C_{\alpha^l_y} + C_{\alpha^r_x} C_{\alpha^r_y}\big) \\
-2\gamma_2 C_{\phi_x}\big(C_{\alpha^l_x} C_{\alpha^l_y} + C_{\alpha^r_x} C_{\alpha^r_y}\big) \\
+\eta_2\big(C^2_{\alpha^l_x} C^2_{\alpha^l_y} + C^2_{\alpha^r_x} C^2_{\alpha^r_y}\big) \\
+2\gamma_2 S_{\phi_x}\big(S_{\alpha^l_x} + S_{\alpha^r_x}\big) \\
-\eta_2\big(C^2_{\alpha^l_x} + C^2_{\alpha^r_x}\big) \\
+\beta + \eta_1 + 2\big(\eta_2 + \eta_5 + \eta_6\big)
\end{array}
&
\begin{array}{c}
\gamma_2 S_{\phi_x} S_{\phi_y}\big(C_{\alpha^l_x} C_{\alpha^l_y} + C_{\alpha^r_x} C_{\alpha^r_y}\big) \\
+\gamma_2 C_{\phi_x} S_{\phi_y}\big(S_{\alpha^l_x} + S_{\alpha^r_x}\big) \\
-\big(\gamma_1 + 2\gamma_3\big) S_{\phi_x} S_{\phi_y} \\
-\eta_3 S_{\phi_x}\big(C_{\alpha^l_x} S_{\alpha^l_y} + C_{\alpha^r_x} S_{\alpha^r_y}\big) \\
+\eta_4 C_{\phi_x}\big(C_{\alpha^l_x} S_{\alpha^l_y} - C_{\alpha^r_x} S_{\alpha^r_y}\big) \\
+\eta_2 S_{\phi_x} C^2_{\alpha^l_x} S_{\alpha^l_y} C_{\alpha^l_y} \\
+\eta_2 S_{\phi_x} C^2_{\alpha^r_x} S_{\alpha^r_y} C_{\alpha^r_y} \\
+\eta_2 C_{\phi_x} S_{\alpha^l_x} C_{\alpha^l_x} S_{\alpha^l_y} \\
+\eta_2 C_{\phi_x} S_{\alpha^r_x} C_{\alpha^r_x} S_{\alpha^r_y}
\end{array}
\\ \hdashline
\begin{array}{c}
\gamma_2 C_{\phi_x} S_{\phi_y}\big(S_{\alpha^l_x} + S_{\alpha^r_x}\big) \\
-\big(\gamma_1 + 2\gamma_3\big) S_{\phi_x} S_{\phi_y} \\
-\eta_3 S_{\phi_x}\big(C_{\alpha^l_x} S_{\alpha^l_y} + C_{\alpha^r_x} S_{\alpha^r_y}\big) \\
+\eta_4 C_{\phi_x}\big(C_{\alpha^l_x} S_{\alpha^l_y} - C_{\alpha^r_x} S_{\alpha^r_y}\big) \\
+\eta_2 S_{\phi_x} C^2_{\alpha^l_x} S_{\alpha^l_y} C_{\alpha^l_y} \\
+\eta_2 S_{\phi_x} C^2_{\alpha^r_x} S_{\alpha^r_y} C_{\alpha^r_y} \\
+\eta_2 S_{\phi_x} C^2_{\alpha^r_x} S_{\alpha^r_y} C_{\alpha^r_y} \\
+\eta_2 C_{\phi_x} S_{\alpha^l_x} C_{\alpha^l_x} S_{\alpha^l_y} \\
+\gamma_2 S_{\phi_x} S_{\phi_y} C_{\alpha^l_x} C_{\alpha^l_y} \\
+\gamma_2 S_{\phi_x} S_{\phi_y} C_{\alpha^r_x} C_{\alpha^r_y}
\end{array}
&
\begin{array}{c}
\beta + 2\eta_4 S^2_{\phi_x}\big(S_{\alpha^l_x} - S_{\alpha^r_y}\big) \\
+2\big(\eta_3 C_{\phi_x} + \eta_2 C_{\phi_y}\big) S_{\phi_x} S_{\alpha^l_x} \\
+2\big(\eta_3 C_{\phi_x} + \eta_2 C_{\phi_y}\big) S_{\phi_x} S_{\alpha^r_x} \\
-\eta_2 S^2_{\phi_x}\big(C^2_{\alpha^l_x} C^2_{\alpha^l_y} + C^2_{\alpha^r_x} C^2_{\alpha^r_y}\big) \\
+\big(2\gamma_2 + 4\gamma_3\big) C_{\phi_x} C_{\phi_y} \\
+2\big(\eta_2 + \eta_6\big) S^2_{\phi_x} + \big(\eta_1 + 2\eta_5\big) C^2_{\phi_x} \\
+\eta_2 C^2_{\phi_x}\big(C^2_{\alpha^l_x} + C^2_{\alpha^r_x}\big) \\
+2\gamma_2 S_{\phi_y}\big(C_{\alpha^l_x} S_{\alpha^l_y} + C_{\alpha^r_x} S_{\alpha^r_y}\big) \\
-2\eta_3 C^2_{\phi_x}\big(C_{\alpha^l_x} C_{\alpha^l_y} + C_{\alpha^r_x} C_{\alpha^r_y}\big) \\
-2\eta_2 S_{\phi_x} C_{\phi_x}\big(S_{\alpha^l_x} C_{\alpha^l_x} C_{\alpha^l_y}\big) \\
-2\eta_2 S_{\phi_x} C_{\phi_x}\big(S_{\alpha^r_x} C_{\alpha^r_x} C_{\alpha^r_y}\big) \\
+2C_{\phi_x}\big(\eta_4 S_{\phi_x} - \gamma_2 C_{\phi_y}\big) C_{\alpha^l_x} C_{\alpha^l_y} \\
+2C_{\phi_x}\big(\eta_4 S_{\phi_x} - \gamma_2 C_{\phi_y}\big) C_{\alpha^r_x} C_{\alpha^r_y}
\end{array}
\end{array}\right]. \quad \text{(A.18)}
$$

The Coriolis and centrifugal matrix $C(q,\dot{q})$ in Eq. A.1 is given by:

$$C(q,\dot{q}) = \begin{bmatrix} \mathbf{0} & C_{\theta\alpha^l}(q_s,\dot{q}_s) & C_{\theta\alpha^r}(q_s,\dot{q}_s) & C_{\theta\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & C_{\alpha^l\alpha^l}(q_s,\dot{q}_s) & \mathbf{0} & C_{\alpha^l\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & \mathbf{0} & C_{\alpha^r\alpha^r}(q_s,\dot{q}_s) & C_{\alpha^r\phi}(q_s,\dot{q}_s) \\ \mathbf{0} & C_{\phi\alpha^l}(q_s,\dot{q}_s) & C_{\phi\alpha^r}(q_s,\dot{q}_s) & C_{\phi\phi}(q_s,\dot{q}_s) \end{bmatrix}, \tag{A.19}$$

where each $C_{ij} \in \mathbb{R}^{2\times 2}$ is given below.

$$C_{\theta\alpha^l}(q,\dot{q}) = \left[\begin{array}{c:c} \begin{array}{c} \gamma_2\big(C_{\phi_x}C_{\alpha_x^l} - S_{\phi_x}S_{\alpha_x^l}C_{\alpha_y^l}\big)S_{\phi_y}\dot{\phi}_x \\ +\gamma_2\big(C_{\phi_x}C_{\phi_y}C_{\alpha_y^l} - S_{\phi_y}S_{\alpha_y^l}\big)S_{\alpha_x^l}\dot{\phi}_y \\ +\gamma_2 S_{\phi_x}C_{\phi_y}C_{\alpha_x^l}\dot{\phi}_y \\ +\gamma_2\big(C_{\phi_x}C_{\alpha_x^l}C_{\alpha_y^l} - S_{\phi_x}S_{\alpha_x^l}\big)S_{\phi_y}\dot{\alpha}_x^l \\ +\gamma_2 C_{\phi_y}C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_x^l \\ +\gamma_2\big(C_{\phi_y}C_{\alpha_y^l} - C_{\phi_x}S_{\phi_y}S_{\alpha_y^l}\big)S_{\alpha_x^l}\dot{\alpha}_y^l \end{array} & \begin{array}{c} -\gamma_2 S_{\phi_x}S_{\phi_y}C_{\alpha_x^l}S_{\alpha_y^l}\dot{\phi}_x \\ +\gamma_2\big(S_{\phi_y}C_{\alpha_y^l} + C_{\phi_x}C_{\phi_y}S_{\alpha_y^l}\big)C_{\alpha_x^l}\dot{\phi}_y \\ +\gamma_2\big(C_{\phi_y}C_{\alpha_y^l} - C_{\phi_x}S_{\phi_y}S_{\alpha_y^l}\big)S_{\alpha_x^l}\dot{\alpha}_x^l \\ +\gamma_2\big(C_{\phi_y}S_{\alpha_y^l} + C_{\phi_x}S_{\phi_y}C_{\alpha_y^l}\big)C_{\alpha_x^l}\dot{\alpha}_y^l \end{array} \\ \hdashline \begin{array}{c} -\gamma_2\big(S_{\phi_x}C_{\alpha_x^l} + C_{\phi_x}S_{\alpha_x^l}C_{\alpha_y^l}\big)\dot{\phi}_x \\ -\gamma_2\big(C_{\phi_x}S_{\alpha_x^l} + S_{\phi_x}C_{\alpha_x^l}C_{\alpha_y^l}\big)\dot{\alpha}_x^l \\ +\gamma_2 S_{\phi_x}S_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l \end{array} & \begin{array}{c} -\gamma_2 C_{\phi_x}C_{\alpha_x^l}S_{\alpha_y^l}\dot{\phi}_x \\ +\gamma_2 S_{\phi_x}S_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_x^l \\ -\gamma_2 S_{\phi_x}C_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_y^l \end{array} \end{array}\right], \tag{A.20}$$

$$C_{\theta\alpha^r}(q,\dot{q}) = \left[\begin{array}{c:c} \begin{array}{c} \gamma_2\big(C_{\phi_x}C_{\alpha_x^r} - S_{\phi_x}S_{\alpha_x^r}C_{\alpha_y^r}\big)S_{\phi_y}\dot{\phi}_x \\ +\gamma_2\big(C_{\phi_x}C_{\phi_y}C_{\alpha_y^r} - S_{\phi_y}S_{\alpha_y^r}\big)S_{\alpha_x^r}\dot{\phi}_y \\ +\gamma_2 S_{\phi_x}C_{\phi_y}C_{\alpha_x^r}\dot{\phi}_y \\ +\gamma_2\big(C_{\phi_x}C_{\alpha_x^r}C_{\alpha_y^r} - S_{\phi_x}S_{\alpha_x^r}\big)S_{\phi_y}\dot{\alpha}_x^r \\ +\gamma_2 C_{\phi_y}C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_x^r \\ +\gamma_2\big(C_{\phi_y}C_{\alpha_y^r} - C_{\phi_x}S_{\phi_y}S_{\alpha_y^r}\big)S_{\alpha_x^r}\dot{\alpha}_y^r \end{array} & \begin{array}{c} -\gamma_2 S_{\phi_x}S_{\phi_y}C_{\alpha_x^r}S_{\alpha_y^r}\dot{\phi}_x \\ +\gamma_2\big(S_{\phi_y}C_{\alpha_y^r} + C_{\phi_x}C_{\phi_y}S_{\alpha_y^r}\big)C_{\alpha_x^r}\dot{\phi}_y \\ +\gamma_2\big(C_{\phi_y}C_{\alpha_y^r} - C_{\phi_x}S_{\phi_y}S_{\alpha_y^r}\big)S_{\alpha_x^r}\dot{\alpha}_x^r \\ +\gamma_2\big(C_{\phi_y}S_{\alpha_y^r} + C_{\phi_x}S_{\phi_y}C_{\alpha_y^r}\big)C_{\alpha_x^r}\dot{\alpha}_y^r \end{array} \\ \hdashline \begin{array}{c} -\gamma_2\big(S_{\phi_x}C_{\alpha_x^r} + C_{\phi_x}S_{\alpha_x^r}C_{\alpha_y^r}\big)\dot{\phi}_x \\ -\gamma_2\big(C_{\phi_x}S_{\alpha_x^r} + S_{\phi_x}C_{\alpha_x^r}C_{\alpha_y^r}\big)\dot{\alpha}_x^r \\ +\gamma_2 S_{\phi_x}S_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r \end{array} & \begin{array}{c} -\gamma_2 C_{\phi_x}C_{\alpha_x^r}S_{\alpha_y^r}\dot{\phi}_x \\ +\gamma_2 S_{\phi_x}S_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_x^r \\ -\gamma_2 S_{\phi_x}C_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_y^r \end{array} \end{array}\right], \tag{A.21}$$

$$C_{\theta\phi}(q,\dot{q}) = \left[\begin{array}{c|c} \begin{array}{c} -\big(\gamma_1+2\gamma_3\big)\big(C_{\phi_x}S_{\phi_y}\dot{\phi}_x + S_{\phi_x}C_{\phi_y}\dot{\phi}_y\big) \\ +\gamma_2 C_{\phi_x}S_{\phi_y}\big(C_{\alpha_x^l}\dot{\alpha}_x^l + C_{\alpha_x^r}\dot{\alpha}_x^r\big) \\ +\gamma_2 S_{\alpha_x^l}\big(C_{\phi_x}C_{\phi_y}\dot{\phi}_y - S_{\phi_x}S_{\phi_y}\dot{\phi}_x\big) \\ +\gamma_2 S_{\alpha_x^r}\big(C_{\phi_x}C_{\phi_y}\dot{\phi}_y - S_{\phi_x}S_{\phi_y}\dot{\phi}_x\big) \\ +\gamma_2 C_{\alpha_x^l}C_{\alpha_y^l}\big(C_{\phi_x}S_{\phi_y}\dot{\phi}_x + S_{\phi_x}C_{\phi_y}\dot{\phi}_y\big) \\ +\gamma_2 C_{\alpha_x^r}C_{\alpha_y^r}\big(C_{\phi_x}S_{\phi_y}\dot{\phi}_x + S_{\phi_x}C_{\phi_y}\dot{\phi}_y\big) \\ -\gamma_2 S_{\phi_x}S_{\phi_y}\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l + C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big) \\ -\gamma_2 S_{\phi_x}S_{\phi_y}\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r + C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big) \end{array} & \begin{array}{c} \gamma_2 S_{\alpha_x^l}\big(C_{\phi_x}C_{\phi_y}\dot{\phi}_x - S_{\phi_x}S_{\phi_y}\dot{\phi}_y\big) \\ +\gamma_2 S_{\alpha_x^r}\big(C_{\phi_x}C_{\phi_y}\dot{\phi}_x - S_{\phi_x}S_{\phi_y}\dot{\phi}_y\big) \\ -\big(\gamma_1+2\gamma_3\big)\big(S_{\phi_x}C_{\phi_y}\dot{\phi}_x + C_{\phi_x}S_{\phi_y}\dot{\phi}_y\big) \\ +\gamma_2 S_{\phi_y}\big(S_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_x^l + S_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_x^r\big) \\ +\gamma_2 S_{\phi_y}\big(C_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_y^l + C_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_y^r\big) \\ +\gamma_2 S_{\phi_x}C_{\phi_y}\big(C_{\alpha_x^l}\dot{\alpha}_x^l + C_{\alpha_x^r}\dot{\alpha}_x^r\big) \\ +\gamma_2 C_{\phi_x}C_{\phi_y}\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l + C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big) \\ +\gamma_2 C_{\phi_x}C_{\phi_y}\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r + C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big) \\ +\gamma_2 C_{\alpha_x^l}C_{\alpha_y^l}\big(S_{\phi_x}C_{\phi_y}\dot{\phi}_x + C_{\phi_x}S_{\phi_y}\dot{\phi}_y\big) \\ +\gamma_2 C_{\alpha_x^r}C_{\alpha_y^r}\big(S_{\phi_x}C_{\phi_y}\dot{\phi}_x + C_{\phi_x}S_{\phi_y}\dot{\phi}_y\big) \\ +\gamma_2\big(C_{\alpha_x^l}S_{\alpha_y^l} + C_{\alpha_x^r}S_{\alpha_y^r}\big)C_{\phi_y}\dot{\phi}_y \end{array} \\ \hline \begin{array}{c} \big(\gamma_1+2\gamma_3\big)S_{\phi_x}\dot{\phi}_x \\ -\gamma_2\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big)C_{\phi_x}\dot{\phi}_x \\ -\gamma_2 S_{\phi_x}\big(C_{\alpha_x^l}\dot{\alpha}_x^l + C_{\alpha_x^r}\dot{\alpha}_x^r\big) \\ -\gamma_2 C_{\phi_x}\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l + C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big) \\ -\gamma_2 C_{\phi_x}\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r + C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big) \\ -\gamma_2\big(C_{\alpha_x^l}C_{\alpha_y^l} + C_{\alpha_x^r}C_{\alpha_y^r}\big)S_{\phi_x}\dot{\phi}_x \end{array} & 0 \end{array}\right], \tag{A.22}$$

$$C_{\alpha^l\alpha^l}(q,\dot{q}) = \left[\begin{array}{c|c} \big(I_a^{zz}-I_a^{xx}\big)S_{\alpha_y^l}C_{\alpha_y^l}\dot{\alpha}_y^l & \begin{array}{c} \big(I_a^{zz}-I_a^{xx}\big)S_{\alpha_y^l}C_{\alpha_y^l}\dot{\alpha}_x^l \\ +\eta_2 S_{\alpha_x^l}C_{\alpha_x^l}\big(C_{\phi_x}\dot{\phi}_y + \dot{\alpha}_y^l\big) \\ +\eta_2 C_{\alpha_x^l}^2\big(S_{\phi_x}C_{\alpha_y^l}\dot{\phi}_y - S_{\alpha_y^l}\dot{\phi}_x\big) \end{array} \\ \hline \begin{array}{c} -\big(I_a^{zz}-I_a^{xx}\big)S_{\alpha_y^l}C_{\alpha_y^l}\dot{\alpha}_x^l \\ -\eta_2 S_{\alpha_x^l}C_{\alpha_x^l}\big(C_{\phi_x}\dot{\phi}_y + \dot{\alpha}_y^l\big) \\ -\eta_2 C_{\alpha_x^l}^2\big(S_{\phi_x}C_{\alpha_y^l}\dot{\phi}_y - S_{\alpha_y^l}\dot{\phi}_x\big) \end{array} & -\eta_2 S_{\alpha_x^l}C_{\alpha_x^l}\dot{\alpha}_x^l \end{array}\right], \tag{A.23}$$

$$C_{\alpha^l\phi}(q,\dot{q}) = \left[\begin{array}{c|c} \begin{array}{c} -\eta_2 S_{\alpha_x^l} C_{\alpha_x^l} S_{\alpha_y^l}^2 \dot{\phi}_x \\ -\eta_2 C_{\phi_x} S_{\alpha_x^l}^2 S_{\alpha_y^l} \dot{\phi}_y \\ -\eta_3 S_{\alpha_x^l} C_{\alpha_x^l} \dot{\phi}_x \\ -\big(\eta_3 - \eta_4\big) S_{\alpha_x^l} S_{\alpha_y^l} \dot{\phi}_y \\ -\eta_3 S_{\alpha_x^l} C_{\alpha_y^l} \dot{\phi}_x \\ +\eta_2 S_{\phi_x} C_{\alpha_x^l} S_{\alpha_y^l} S_{\alpha_x^l} C_{\alpha_y^l} \dot{\phi}_y \\ -\eta_2 C_{\alpha_x^l}^2 S_{\alpha_y^l} \dot{\alpha}_y^l - \eta_4 C_{\alpha_x^l} \dot{\phi}_x \end{array} & \begin{array}{c} \big(\eta_3 C_{\phi_x} - \eta_4 S_{\phi_x}\big) S_{\phi_x} C_{\alpha_x^l} \dot{\phi}_y \\ +\eta_2 C_{\phi_x} \big(S_{\alpha_y^l} \dot{\phi}_x + C_{\alpha_x^l} \dot{\alpha}_y^l\big) \\ +\big(\eta_4 C_{\phi_x} - \eta_3 S_{\phi_x}\big) S_{\alpha_x^l} S_{\alpha_y^l} \dot{\phi}_x \\ +\eta_2 S_{\phi_x} C_{\alpha_y^l} \big(C_{\alpha_x^l}^2 \dot{\alpha}_y^l - C_{\phi_x} \dot{\phi}_y\big) \\ -\big(\eta_2 S_{\alpha_x^l} C_{\alpha_y^l} + \eta_3 C_{\phi_x}^2\big) C_{\alpha_x^l} C_{\alpha_y^l} \dot{\phi}_y \\ \eta_2 \big(S_{\phi_x} S_{\alpha_x^l} C_{\alpha_y^l} - C_{\phi_x} C_{\alpha_x^l}\big) C_{\alpha_x^l} S_{\alpha_y^l} \dot{\phi}_x \\ +\eta_2 C_{\phi_x}^2 S_{\alpha_x^l} C_{\alpha_x^l} \dot{\phi}_y \big(1 + C_{\alpha_y^l}^2\big) \\ \big(2\eta_2 C_{\alpha_x^l}^2 - \eta_4 S_{\alpha_x^l}\big) S_{\phi_x} C_{\phi_x} C_{\alpha_y^l} \dot{\phi}_y \end{array} \\ \hline \begin{array}{c} -\eta_3 C_{\alpha_x^l} S_{\alpha_y^l} \dot{\phi}_x \\ +\eta_3 S_{\phi_x} C_{\alpha_x^l} C_{\alpha_y^l} \dot{\phi}_y \\ -\eta_4 C_{\phi_x} C_{\alpha_x^l} C_{\alpha_y^l} \dot{\phi}_y \\ -\eta_2 S_{\phi_x} C_{\alpha_x^l}^2 C_{\alpha_y^l}^2 \dot{\phi}_y \\ -\eta_2 C_{\phi_x} S_{\alpha_x^l} C_{\alpha_x^l} C_{\alpha_y^l} \dot{\phi}_y \\ +\eta_2 C_{\alpha_x^l}^2 S_{\alpha_y^l} \big(\dot{\alpha}_x^l + C_{\alpha_y^l} \dot{\phi}_x\big) \end{array} & \begin{array}{c} -\big(\eta_3 C_{\phi_x} + \eta_4 S_{\phi_x}\big) C_{\alpha_x^l} S_{\alpha_y^l} C_{\phi_x} \dot{\phi}_y \\ -\eta_2 \big(S_{\alpha_x^l} C_{\phi_x} + C_{\alpha_x^l} C_{\alpha_y^l} S_{\phi_x}\big) C_{\alpha_x^l} \dot{\alpha}_x^l \\ +\big(\eta_3 S_{\phi_x} - \eta_4 C_{\phi_x}\big) C_{\alpha_x^l} C_{\alpha_y^l} \dot{\phi}_x \\ -\eta_2 C_{\alpha_x^l}^2 S_{\alpha_y^l} C_{\alpha_y^l} S_{\phi_x}^2 \dot{\phi}_y \\ -\eta_2 C_{\alpha_x^l} S_{\alpha_x^l} S_{\alpha_y^l} S_{\phi_x} C_{\phi_x} \dot{\phi}_y \\ -\eta_2 C_{\alpha_x^l}^2 C_{\alpha_y^l}^2 S_{\phi_x} \dot{\phi}_x \\ -\eta_2 S_{\alpha_x^l} C_{\alpha_x^l} C_{\alpha_y^l} C_{\phi_x} \dot{\phi}_x \end{array} \end{array}\right] \tag{A.24}$$

$$C_{\alpha^r\alpha^r}(q,\dot{q}) = \left[\begin{array}{c|c} \big(I_a^{zz} - I_a^{xx}\big) S_{\alpha_y^r} C_{\alpha_y^r} \dot{\alpha}_y^r & \begin{array}{c} \big(I_a^{zz} - I_a^{xx}\big) S_{\alpha_y^r} C_{\alpha_y^r} \dot{\alpha}_x^r \\ +\eta_2 S_{\alpha_x^r} C_{\alpha_x^r} \big(C_{\phi_x} \dot{\phi}_y + \dot{\alpha}_y^r\big) \\ +\eta_2 C_{\alpha_x^r}^2 \big(S_{\phi_x} C_{\alpha_y^r} \dot{\phi}_y - S_{\alpha_y^r} \dot{\phi}_x\big) \end{array} \\ \hline \begin{array}{c} -\big(I_a^{zz} - I_a^{xx}\big) S_{\alpha_y^r} C_{\alpha_y^r} \dot{\alpha}_x^r \\ -\eta_2 S_{\alpha_x^r} C_{\alpha_x^r} \big(C_{\phi_x} \dot{\phi}_y + \dot{\alpha}_y^r\big) \\ -\eta_2 C_{\alpha_x^r}^2 \big(S_{\phi_x} C_{\alpha_y^r} \dot{\phi}_y - S_{\alpha_y^r} \dot{\phi}_x\big) \end{array} & -\eta_2 S_{\alpha_x^r} C_{\alpha_x^r} \dot{\alpha}_x^r \end{array}\right], \tag{A.25}$$

$$C_{\alpha^r\phi}(q,\dot{q}) = \left[\begin{array}{c|c} \begin{array}{c} -\eta_2 S_{\alpha_x^r} C_{\alpha_x^r} S_{\alpha_y^r}^2 \dot{\phi}_x \\ -\eta_2 C_{\phi_x} S_{\alpha_x^r}^2 S_{\alpha_y^r} \dot{\phi}_y \\ -\eta_3 S_{\alpha_x^r} C_{\alpha_x^r} \dot{\phi}_x \\ -\big(\eta_3 - \eta_4\big) S_{\alpha_x^r} S_{\alpha_y^r} \dot{\phi}_y \\ -\eta_3 S_{\alpha_x^r} C_{\alpha_y^r} \dot{\phi}_x \\ +\eta_2 S_{\phi_x} C_{\alpha_x^r} S_{\alpha_y^r} S_{\alpha_x^r} C_{\alpha_y^r} \dot{\phi}_y \\ -\eta_2 C_{\alpha_x^r}^2 S_{\alpha_y^r} \dot{\alpha}_y^r - \eta_4 C_{\alpha_x^r} \dot{\phi}_x \end{array} & \begin{array}{c} \big(\eta_3 C_{\phi_x} - \eta_4 S_{\phi_x}\big) S_{\phi_x} C_{\alpha_x^r} \dot{\phi}_y \\ +\eta_2 C_{\phi_x} \big(S_{\alpha_y^r} \dot{\phi}_x + C_{\alpha_x^r} \dot{\alpha}_y^r\big) \\ +\big(\eta_4 C_{\phi_x} - \eta_3 S_{\phi_x}\big) S_{\alpha_x^r} S_{\alpha_y^r} \dot{\phi}_x \\ +\eta_2 S_{\phi_x} C_{\alpha_y^r} \big(C_{\alpha_x^r}^2 \dot{\alpha}_y^r - C_{\phi_x} \dot{\phi}_y\big) \\ -\big(\eta_2 S_{\alpha_x^r} C_{\alpha_y^r} + \eta_3 C_{\phi_x}^2\big) C_{\alpha_x^r} C_{\alpha_y^r} \dot{\phi}_y \\ \eta_2 \big(S_{\phi_x} S_{\alpha_x^r} C_{\alpha_y^r} - C_{\phi_x} C_{\alpha_x^r}\big) C_{\alpha_x^r} S_{\alpha_y^r} \dot{\phi}_x \\ +\eta_2 C_{\phi_x}^2 S_{\alpha_x^r} C_{\alpha_x^r} \dot{\phi}_y \big(1 + C_{\alpha_y^r}^2\big) \\ \big(2\eta_2 C_{\alpha_x^r}^2 - \eta_4 S_{\alpha_x^r}\big) S_{\phi_x} C_{\phi_x} C_{\alpha_y^r} \dot{\phi}_y \end{array} \\ \hline \begin{array}{c} -\eta_3 C_{\alpha_x^r} S_{\alpha_y^r} \dot{\phi}_x \\ +\eta_3 S_{\phi_x} C_{\alpha_x^r} C_{\alpha_y^r} \dot{\phi}_y \\ -\eta_4 C_{\phi_x} C_{\alpha_x^r} C_{\alpha_y^r} \dot{\phi}_y \\ -\eta_2 S_{\phi_x} C_{\alpha_x^r}^2 C_{\alpha_y^r}^2 \dot{\phi}_y \\ -\eta_2 C_{\phi_x} S_{\alpha_x^r} C_{\alpha_x^r} C_{\alpha_y^r} \dot{\phi}_y \\ +\eta_2 C_{\alpha_x^r}^2 S_{\alpha_y^r} \big(\dot{\alpha}_x^r + C_{\alpha_y^r} \dot{\phi}_x\big) \end{array} & \begin{array}{c} -\big(\eta_3 C_{\phi_x} + \eta_4 S_{\phi_x}\big) C_{\alpha_x^r} S_{\alpha_y^r} C_{\phi_x} \dot{\phi}_y \\ -\eta_2 \big(S_{\alpha_x^r} C_{\phi_x} + C_{\alpha_x^r} C_{\alpha_y^r} S_{\phi_x}\big) C_{\alpha_x^r} \dot{\alpha}_x^r \\ +\big(\eta_3 S_{\phi_x} - \eta_4 C_{\phi_x}\big) C_{\alpha_x^r} C_{\alpha_y^r} \dot{\phi}_x \\ -\eta_2 C_{\alpha_x^r}^2 S_{\alpha_y^r} C_{\alpha_y^r} S_{\phi_x}^2 \dot{\phi}_y \\ -\eta_2 C_{\alpha_x^r} S_{\alpha_x^r} S_{\alpha_y^r} S_{\phi_x} C_{\phi_x} \dot{\phi}_y \\ -\eta_2 C_{\alpha_x^r}^2 C_{\alpha_y^r}^2 S_{\phi_x} \dot{\phi}_x \\ -\eta_2 S_{\alpha_x^r} C_{\alpha_x^r} C_{\alpha_y^r} C_{\phi_x} \dot{\phi}_x \end{array} \end{array}\right], \quad \text{(A.26)}$$

$$
C_{\phi\alpha_l}(q,\dot{q}) = \left[\begin{array}{c|c}
\begin{array}{c}
\eta_4 C_{\alpha_x^l}\big(\dot{\phi}_x + C_{\alpha_y^l}\dot{\alpha}_x^l\big) \\
-\eta_4 S_{\alpha_x^l} S_{\alpha_y^l}\big(\dot{\alpha}_y^l + C_{\phi_x}\dot{\phi}_y\big) \\
+\eta_3 S_{\alpha_x^l}\big(\dot{\alpha}_x^l + C_{\alpha_y^l}\dot{\phi}_x\big) \\
+\big(\eta_3 S_{\alpha_x^l} S_{\phi_x} - \eta_2 C_{\phi_x}\big) S_{\alpha_y^l}\dot{\phi}_y \\
+\eta_2\big(\dot{\alpha}_y^l + C_{\phi_x}\dot{\phi}_y\big) C_{\alpha_x^l}^2 S_{\alpha_y^l} \\
+\eta_2\big(S_{\alpha_y^l}\dot{\phi}_x - C_{\alpha_y^l} S_{\phi_x}\dot{\phi}_y\big) S_{\alpha_x^l} C_{\alpha_x^l} S_{\alpha_y^l} \\
+\gamma_2\big(\dot{\alpha}_x^l + C_{\alpha_y^l}\dot{\phi}_x\big) C_{\phi_x} S_{\alpha_x^l} \\
+\gamma_2\big(C_{\alpha_y^l}\dot{\alpha}_x^l + \dot{\phi}_x\big) S_{\phi_x} C_{\alpha_x^l} \\
-\big(\eta_2 + \gamma_2 S_{\alpha_x^l} S_{\phi_x}\big) S_{\alpha_y^l}\dot{\alpha}_y^l
\end{array} &
\begin{array}{c}
-\eta_2 S_{\alpha_x^l}^2 S_{\alpha_y^l}\dot{\alpha}_x^l \\
+\eta_2 C_{\alpha_x^l}^2 C_{\alpha_y^l}^2 S_{\phi_x}\dot{\phi}_y \\
+\eta_2\big(\dot{\alpha}_y^l + C_{\phi_x}\dot{\phi}_y\big) S_{\alpha_x^l} C_{\alpha_x^l} C_{\alpha_y^l} \\
-\eta_2 C_{\alpha_x^l}^2 C_{\alpha_y^l} S_{\alpha_y^l}\dot{\phi}_x \\
+\eta_4\big(\dot{\alpha}_y^l + C_{\phi_x}\dot{\phi}_y\big) C_{\alpha_x^l} C_{\alpha_y^l} \\
-\eta_4 S_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_x^l \\
+\eta_3 C_{\alpha_x^l}\big(S_{\alpha_y^l}\dot{\phi}_x - C_{\alpha_y^l} S_{\phi_x}\dot{\phi}_y\big) \\
+\gamma_2 C_{\alpha_x^l}\big(C_{\alpha_y^l} S_{\phi_x}\dot{\alpha}_y^l + S_{\alpha_y^l} C_{\phi_x}\dot{\phi}_x\big) \\
-\gamma_2 S_{\alpha_x^l} S_{\alpha_y^l} S_{\phi_x}\dot{\alpha}_x^l
\end{array} \\
\hline
\begin{array}{c}
\eta_4 C_{\alpha_x^l} S_{\phi_x}\big(S_{\phi_x}\dot{\phi}_y + S_{\alpha_y^l}\dot{\alpha}_x^l\big) \\
+\eta_4 S_{\alpha_x^l} C_{\alpha_y^l} S_{\phi_x}\big(C_{\phi_x}\dot{\phi}_y + \dot{\alpha}_y^l\big) \\
+\eta_3 C_{\alpha_x^l} C_{\phi_x}\big(S_{\phi_x}\dot{\phi}_y + S_{\alpha_y^l}\dot{\alpha}_x^l\big) \\
+\eta_3 S_{\alpha_x^l} C_{\alpha_y^l} C_{\phi_x}\big(C_{\phi_x}\dot{\phi}_y + \dot{\alpha}_y^l\big) \\
+\eta_2 S_{\alpha_x^l} C_{\alpha_x^l} C_{\phi_x}^2\dot{\phi}_y\big(1 + C_{\alpha_y^l}^2\big) \\
+\eta_2 C_{\alpha_x^l} C_{\alpha_y^l}\dot{\phi}_y\big(S_{\alpha_x^l} C_{\alpha_y^l} - 2C_{\alpha_x^l} S_{\phi_x} C_{\phi_x}\big) \\
+\eta_2 S_{\phi_x} C_{\alpha_y^l}\big(\dot{\alpha}_y^l + C_{\phi_x}\dot{\phi}_y\big) \\
+\eta_2 C_{\phi_x} C_{\alpha_x^l}\big(C_{\alpha_x^l} S_{\alpha_y^l}\dot{\phi}_x - S_{\alpha_x^l}\dot{\alpha}_y^l\big) \\
-\eta_2 S_{\phi_x} C_{\alpha_x^l} C_{\alpha_y^l}\big(C_{\alpha_x^l}\dot{\alpha}_y^l + S_{\alpha_x^l} S_{\alpha_y^l}\dot{\phi}_x\big) \\
+\gamma_2 C_{\phi_y}\big(C_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_x^l + S_{\alpha_x^l} C_{\alpha_y^l}\dot{\alpha}_y^l\big) \\
+\gamma_2 C_{\phi_x} S_{\phi_y} C_{\alpha_x^l}\big(\dot{\phi}_x + C_{\alpha_y^l}\dot{\alpha}_x^l\big) \\
+\gamma_2 S_{\phi_x} C_{\phi_y} C_{\alpha_x^l}\dot{\phi}_y \\
-\gamma_2 S_{\phi_y} S_{\alpha_x^l}\big(S_{\alpha_y^l}\dot{\phi}_y + S_{\phi_x}\dot{\alpha}_x^l\big) \\
+\gamma_2 S_{\alpha_x^l} C_{\alpha_y^l}\big(C_{\phi_x} C_{\phi_y}\dot{\phi}_y - S_{\phi_x} S_{\phi_y}\dot{\phi}_x\big) \\
-\gamma_2 C_{\phi_x} S_{\phi_y} S_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_y^l
\end{array} &
\begin{array}{c}
\gamma_2 C_{\phi_y} S_{\alpha_x^l} C_{\alpha_y^l}\dot{\alpha}_x^l \\
+\big(\eta_3 C_{\phi_x} + \eta_4 S_{\phi_x}\big) S_{\alpha_x^l} C_{\alpha_y^l}\dot{\alpha}_x^l \\
+\big(\eta_3 C_{\phi_x} + \eta_4 S_{\phi_x}\big) C_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_y^l \\
+\big(\eta_3 C_{\phi_x} + \eta_4 S_{\phi_x}\big) C_{\alpha_x^l} S_{\alpha_y^l} C_{\phi_x}\dot{\phi}_y \\
-\eta_2 S_{\phi_x}\big(C_{\alpha_x^l}^2 S_{\alpha_y^l}^2\dot{\phi}_x + C_{\alpha_y^l}\dot{\alpha}_x^l\big) \\
+\eta_2 S_{\phi_x} C_{\alpha_x^l} S_{\alpha_x^l} S_{\alpha_y^l}\big(\dot{\alpha}_y^l + C_{\phi_x}\dot{\phi}_y\big) \\
-\eta_2 C_{\alpha_x^l}^2 C_{\alpha_y^l}\big(S_{\phi_x}\dot{\alpha}_x^l + C_{\phi_x}^2 S_{\alpha_y^l}\dot{\phi}_y\big) \\
+\eta_2 C_{\alpha_x^l}^2 C_{\alpha_y^l} S_{\alpha_y^l}\dot{\phi}_y \\
-\eta_2 C_{\phi_x} S_{\alpha_x^l} C_{\alpha_x^l}\dot{\alpha}_x^l \\
+\gamma_2 C_{\alpha_x^l} S_{\phi_y}\big(C_{\phi_x} C_{\alpha_y^l}\dot{\alpha}_y^l - S_{\phi_x} S_{\alpha_y^l}\dot{\phi}_x\big) \\
+\gamma_2 C_{\alpha_x^l} S_{\alpha_y^l} C_{\phi_y}\big(\dot{\alpha}_y^l + C_{\phi_x}\dot{\phi}_y\big) \\
+\gamma_2 S_{\phi_y}\big(C_{\alpha_x^l} C_{\alpha_y^l}\dot{\phi}_y - C_{\phi_x} S_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_x^l\big)
\end{array}
\end{array}\right], \tag{A.27}
$$

$$
C_{\phi\alpha_r}(q,\dot{q}) = \left[\begin{array}{c|c}
\begin{array}{c}
-\eta_4 C_{\alpha_x^r}\big(\dot{\phi}_x + C_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
+\eta_4 S_{\alpha_x^r} S_{\alpha_y^r}\big(\dot{\alpha}_y^r + C_{\phi_x}\dot{\phi}_y\big) \\
+\eta_3 S_{\alpha_x^r}\big(\dot{\alpha}_x^r + C_{\alpha_y^r}\dot{\phi}_x\big) \\
+\big(\eta_3 S_{\alpha_x^r} S_{\phi_x} - \eta_2 C_{\phi_x}\big) S_{\alpha_y^r}\dot{\phi}_y \\
+\eta_2\big(\dot{\alpha}_y^r + C_{\phi_x}\dot{\phi}_y\big) C_{\alpha_x^r}^2 S_{\alpha_y^r} \\
+\eta_2\big(S_{\alpha_y^r}\dot{\phi}_x - C_{\alpha_y^r} S_{\phi_x}\dot{\phi}_y\big) S_{\alpha_x^r} C_{\alpha_x^r} S_{\alpha_y^r} \\
+\gamma_2\big(\dot{\alpha}_x^r + C_{\alpha_y^r}\dot{\phi}_x\big) C_{\phi_x} S_{\alpha_x^r} \\
+\gamma_2\big(C_{\alpha_y^r}\dot{\alpha}_x^r + \dot{\phi}_x\big) S_{\phi_x} C_{\alpha_x^r} \\
-\big(\eta_2 + \gamma_2 S_{\alpha_x^r} S_{\phi_x}\big) S_{\alpha_y^r}\dot{\alpha}_y^r
\end{array} &
\begin{array}{c}
-\eta_2 S_{\alpha_x^r}^2 S_{\alpha_y^r}\dot{\alpha}_x^r \\
+\eta_2 C_{\alpha_x^r}^2 C_{\alpha_y^r}^2 S_{\phi_x}\dot{\phi}_y \\
+\eta_2\big(\dot{\alpha}_y^r + C_{\phi_x}\dot{\phi}_y\big) S_{\alpha_x^r} C_{\alpha_x^r} C_{\alpha_y^r} \\
-\eta_2 C_{\alpha_x^r}^2 C_{\alpha_y^r} S_{\alpha_y^r}\dot{\phi}_x \\
-\eta_4\big(\dot{\alpha}_y^r + C_{\phi_x}\dot{\phi}_y\big) C_{\alpha_x^r} C_{\alpha_y^r} \\
+\eta_4 S_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_x^r \\
+\eta_3 C_{\alpha_x^r}\big(S_{\alpha_y^r}\dot{\phi}_x - C_{\alpha_y^r} S_{\phi_x}\dot{\phi}_y\big) \\
+\gamma_2 C_{\alpha_x^r}\big(C_{\alpha_y^r} S_{\phi_x}\dot{\alpha}_y^r + S_{\alpha_y^r} C_{\phi_x}\dot{\phi}_x\big) \\
-\gamma_2 S_{\alpha_x^r} S_{\alpha_y^r} S_{\phi_x}\dot{\alpha}_x^r
\end{array} \\
\hline
\begin{array}{c}
-\eta_4 C_{\alpha_x^r} S_{\phi_x}\big(S_{\phi_x}\dot{\phi}_y + S_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
-\eta_4 S_{\alpha_x^r} C_{\alpha_y^r} S_{\phi_x}\big(C_{\phi_x}\dot{\phi}_y + \dot{\alpha}_y^r\big) \\
+\eta_3 C_{\alpha_x^r} C_{\phi_x}\big(S_{\phi_x}\dot{\phi}_y + S_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
+\eta_3 S_{\alpha_x^r} C_{\alpha_y^r} C_{\phi_x}\big(C_{\phi_x}\dot{\phi}_y + \dot{\alpha}_y^r\big) \\
+\eta_2 S_{\alpha_x^r} C_{\alpha_x^r} C_{\phi_x}^2\dot{\phi}_y\big(1 + C_{\alpha_y^r}^2\big) \\
+\eta_2 C_{\alpha_x^r} C_{\alpha_y^r}\dot{\phi}_y\big(S_{\alpha_x^r} C_{\alpha_y^r} - 2C_{\alpha_x^r} S_{\phi_x} C_{\phi_x}\big) \\
+\eta_2 S_{\phi_x} C_{\alpha_y^r}\big(\dot{\alpha}_y^r + C_{\phi_x}\dot{\phi}_y\big) \\
+\eta_2 C_{\phi_x} C_{\alpha_x^r}\big(C_{\alpha_x^r} S_{\alpha_y^r}\dot{\phi}_x - S_{\alpha_x^r}\dot{\alpha}_y^r\big) \\
-\eta_2 S_{\phi_x} C_{\alpha_x^r} C_{\alpha_y^r}\big(C_{\alpha_x^r}\dot{\alpha}_y^r + S_{\alpha_x^r} S_{\alpha_y^r}\dot{\phi}_x\big) \\
+\gamma_2 C_{\phi_y}\big(C_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_x^r + S_{\alpha_x^r} C_{\alpha_y^r}\dot{\alpha}_y^r\big) \\
+\gamma_2 C_{\phi_x} S_{\phi_y} C_{\alpha_x^r}\big(\dot{\phi}_x + C_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
+\gamma_2 S_{\phi_x} C_{\phi_y} C_{\alpha_x^r}\dot{\phi}_y \\
-\gamma_2 S_{\phi_y} S_{\alpha_x^r}\big(S_{\alpha_y^r}\dot{\phi}_y + S_{\phi_x}\dot{\alpha}_x^r\big) \\
+\gamma_2 S_{\alpha_x^r} C_{\alpha_y^r}\big(C_{\phi_x} C_{\phi_y}\dot{\phi}_y - S_{\phi_x} S_{\phi_y}\dot{\phi}_x\big) \\
-\gamma_2 C_{\phi_x} S_{\phi_y} S_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_y^r
\end{array} &
\begin{array}{c}
\gamma_2 C_{\phi_y} S_{\alpha_x^r} C_{\alpha_y^r}\dot{\alpha}_x^r \\
+\big(\eta_3 C_{\phi_x} - \eta_4 S_{\phi_x}\big) S_{\alpha_x^r} C_{\alpha_y^r}\dot{\alpha}_x^r \\
+\big(\eta_3 C_{\phi_x} - \eta_4 S_{\phi_x}\big) C_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_y^r \\
+\big(\eta_3 C_{\phi_x} - \eta_4 S_{\phi_x}\big) C_{\alpha_x^r} S_{\alpha_y^r} C_{\phi_x}\dot{\phi}_y \\
-\eta_2 S_{\phi_x}\big(C_{\alpha_x^r}^2 S_{\alpha_y^r}^2\dot{\phi}_x + C_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
+\eta_2 S_{\phi_x} C_{\alpha_x^r} S_{\alpha_x^r} S_{\alpha_y^r}\big(\dot{\alpha}_y^r + C_{\phi_x}\dot{\phi}_y\big) \\
-\eta_2 C_{\alpha_x^r}^2 C_{\alpha_y^r}\big(S_{\phi_x}\dot{\alpha}_x^r + C_{\phi_x}^2 S_{\alpha_y^r}\dot{\phi}_y\big) \\
+\eta_2 C_{\alpha_x^r}^2 C_{\alpha_y^r} S_{\alpha_y^r}\dot{\phi}_y \\
-\eta_2 C_{\phi_x} S_{\alpha_x^r} C_{\alpha_x^r}\dot{\alpha}_x^r \\
+\gamma_2 C_{\alpha_x^r} S_{\phi_y}\big(C_{\phi_x} C_{\alpha_y^r}\dot{\alpha}_y^r - S_{\phi_x} S_{\alpha_y^r}\dot{\phi}_x\big) \\
+\gamma_2 C_{\alpha_x^r} S_{\alpha_y^r} C_{\phi_y}\big(\dot{\alpha}_y^r + C_{\phi_x}\dot{\phi}_y\big) \\
+\gamma_2 S_{\phi_y}\big(C_{\alpha_x^r} C_{\alpha_y^r}\dot{\phi}_y - C_{\phi_x} S_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_x^r\big)
\end{array}
\end{array}\right], \tag{A.28}
$$

$$
C_{\phi\phi}(q,\dot{q}) = \left[\begin{array}{c:c}
\begin{array}{c}
\big(\eta_3 - \eta_2 C_{\alpha_x^l} C_{\alpha_y^l}\big) S_{\alpha_x^l} C_{\alpha_y^l} \dot{\alpha}_x^l \\
+\big(\eta_3 - \eta_2 C_{\alpha_x^l} C_{\alpha_y^l}\big) C_{\alpha_x^l} S_{\alpha_y^l} \dot{\alpha}_y^l \\
+\big(\eta_3 - \eta_2 C_{\alpha_x^r} C_{\alpha_y^r}\big) S_{\alpha_x^r} C_{\alpha_y^r} \dot{\alpha}_x^r \\
+\big(\eta_3 - \eta_2 C_{\alpha_x^r} C_{\alpha_y^r}\big) C_{\alpha_x^r} S_{\alpha_y^r} \dot{\alpha}_y^r \\
+\gamma_2 S_{\phi_x}\big(C_{\alpha_x^l}\dot{\alpha}_x^l + C_{\alpha_x^r}\dot{\alpha}_x^r\big) \\
+\gamma_2 C_{\phi_x}\big(S_{\alpha_x^l} C_{\alpha_y^l}\dot{\alpha}_x^l + C_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_y^l\big) \\
+\gamma_2\big(C_{\alpha_x^l} C_{\alpha_y^l} + C_{\alpha_x^r} C_{\alpha_y^r}\big) S_{\phi_x}\dot{\phi}_x \\
+\gamma_2 C_{\phi_x}\big(S_{\alpha_x^r} C_{\alpha_y^r}\dot{\alpha}_x^r + C_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_y^r\big) \\
+\big(I_{zz}^b - I_{xx}^b\big) S_{\phi_y} C_{\phi_y}\dot{\phi}_y \\
+\big(\eta_4 + \eta_2 S_{\alpha_x^l}\big) C_{\alpha_x^l}\dot{\alpha}_x^l \\
-\big(\gamma_1 + 2\gamma_3\big) S_{\phi_x}\dot{\phi}_x \\
-\big(\eta_4 - \eta_2 S_{\alpha_x^r}\big) C_{\alpha_x^r}\dot{\alpha}_x^r \\
+\gamma_2 C_{\phi_x}\dot{\phi}_x\big(S_{\alpha_x^l} + S_{\alpha_x^r}\big)
\end{array} &
\begin{array}{c}
\big(I_{zz}^b - I_{xx}^b\big) S_{\phi_y} C_{\phi_y}\dot{\phi}_x \\
-\eta_2 C_{\phi_x}\big(S_{\alpha_y^l}\dot{\alpha}_x^l + S_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
+S_{2\phi_x}\big(\eta_5 - \eta_6\big) - \eta_2 S_{2\phi_x}\dot{\phi}_y \\
+\eta_1 S_{\phi_x} C_{\phi_x}\dot{\phi}_y + \eta_3\dot{\phi}_y\big(S_{\alpha_x^l} + S_{\alpha_x^r}\big) \\
+\eta_2 S_{\phi_x}\big(C_{\alpha_x^l}^2 C_{\alpha_y^l}^2\dot{\alpha}_y^l + C_{\alpha_x^r}^2 C_{\alpha_y^r}^2\dot{\alpha}_y^r\big) \\
-\eta_2\dot{\phi}_y\big(C_{\alpha_x^l} C_{\alpha_y^l} S_{\alpha_x^l} + C_{\alpha_x^r} C_{\alpha_y^r} S_{\alpha_x^r}\big) \\
+\eta_4\dot{\phi}_y\big(C_{\alpha_x^r} C_{\alpha_y^r} - C_{\alpha_x^l} C_{\alpha_y^l}\big) \\
+\eta_2 C_{\phi_x}\big(C_{\alpha_x^l}^2 S_{\alpha_y^l}\dot{\alpha}_x^l + C_{\alpha_x^r}^2 S_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
+\eta_2 S_{\phi_x} C_{\phi_x}\dot{\phi}_y\big(C_{\alpha_x^l}^2 + C_{\alpha_x^r}^2\big) \\
-2\eta_3 C_{\phi_x}^2\dot{\phi}_y\big(S_{\alpha_x^l} + S_{\alpha_x^r}\big) \\
+\eta_2 C_{\phi_x}\big(S_{\alpha_x^l} C_{\alpha_x^l} C_{\alpha_y^l} + S_{\alpha_x^r} C_{\alpha_x^r} C_{\alpha_y^r}\big) \\
+\eta_4 C_{\phi_x}\big(C_{\alpha_x^l} C_{\alpha_y^l}\dot{\alpha}_y^l - C_{\alpha_x^r} C_{\alpha_y^r}\dot{\alpha}_y^r\big) \\
-\eta_3 S_{\phi_x}\big(C_{\alpha_x^l} C_{\alpha_y^l}\dot{\alpha}_y^l + C_{\alpha_x^r} C_{\alpha_y^r}\dot{\alpha}_y^r\big) \\
-\eta_4 C_{\phi_x}\big(S_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_x^l - S_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
+\eta_2 C_{\phi_x}^2\dot{\phi}_y\big(S_{2\alpha_x^l} C_{\alpha_y^l} + S_{2\alpha_x^r} C_{\alpha_y^r}\big) \\
+\eta_3 S_{\phi_x}\big(S_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_x^l + S_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_x^r\big) \\
+\eta_4 S_{2\phi_x}\dot{\phi}_y\big(S_{\alpha_x^r} - S_{\alpha_x^l}\big) \\
+2\big(\eta_4 C_{\phi_x} - \eta_3 S_{\phi_x}\big) C_{\alpha_x^l} C_{\alpha_y^l} C_{\phi_x}\dot{\phi}_y \\
-2\big(\eta_4 C_{\phi_x} + \eta_3 S_{\phi_x}\big) C_{\alpha_x^r} C_{\alpha_y^r} C_{\phi_x}\dot{\phi}_y \\
+\eta_2\big(C_{\alpha_x^l}^2 C_{\alpha_y^l}^2 + C_{\alpha_x^r}^2 C_{\alpha_y^r}^2\big) S_{\phi_x} C_{\phi_x}\dot{\phi}_y \\
-\eta_2 S_{\phi_x} C_{\alpha_x^l} C_{\alpha_y^l} S_{\alpha_x^l} S_{\alpha_y^l}\dot{\alpha}_x^l \\
-\eta_2 S_{\phi_x} C_{\alpha_x^r} C_{\alpha_y^r} S_{\alpha_x^r} S_{\alpha_y^r}\dot{\alpha}_x^r
\end{array} \\
\hdashline
C_{\phi\phi}^{21}(q,\dot{q}) & C_{\phi\phi}^{22}(q,\dot{q})
\end{array}\right] . \tag{A.29}
$$

The contents of the second row elements of $C_{\phi\phi}(q,\dot{q})$ are given below.

$$
\begin{aligned}
C_{\phi\phi}^{21}(q,\dot{q}) \;=\; & \big(I_{xx}^b - I_{zz}^b\big) S_{\phi_y} C_{\phi_y}\dot{\phi}_x + \big(\eta_6 - \eta_5 + \eta_2 - \frac{1}{2}\eta_1\big) S_{2\phi_x}\dot{\phi}_y - \eta_3\big(S_{\alpha_x^l} + S_{\alpha_x^r}\big)\dot{\phi}_y \\
& -\eta_2\big(C_{\alpha_x^l}^2 S_{\alpha_y^l}^2 S_{\phi_x}\dot{\alpha}_y^l + C_{\alpha_x^r}^2 S_{\alpha_y^r}^2 S_{\phi_x}\dot{\alpha}_y^r\big) + \eta_4\big(C_{\alpha_x^l} C_{\alpha_y^l} - C_{\alpha_x^r} C_{\alpha_y^r}\big)\dot{\phi}_y
\end{aligned}
$$

$$
\begin{aligned}
&-(\gamma_1+2\gamma_3)\big(C_{\phi_x}S_{\phi_y}\dot{\phi}_x+S_{\phi_x}C_{\phi_y}\dot{\phi}_y\big)+2\eta_3C_{\phi_x}^2\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big)\dot{\phi}_y\\
&+\eta_2\big(S_{\alpha_x^l}C_{\alpha_x^l}C_{\alpha_y^l}+S_{\alpha_x^r}C_{\alpha_x^r}C_{\alpha_y^r}\big)\dot{\phi}_y-\eta_3\big(C_{\alpha_x^l}S_{\alpha_y^l}+C_{\alpha_x^r}S_{\alpha_y^r}\big)C_{\phi_x}\dot{\phi}_x\\
&-\eta_2\big(S_{\alpha_x^l}C_{\alpha_x^l}S_{\alpha_y^l}+S_{\alpha_x^r}C_{\alpha_x^r}S_{\alpha_y^r}\big)S_{\phi_x}\dot{\phi}_x-\eta_2\big(C_{\alpha_x^l}^2+C_{\alpha_x^r}^2\big)S_{\phi_x}C_{\phi_x}\dot{\phi}_y\\
&+\eta_2C_{\phi_x}\big(C_{\alpha_x^l}^2S_{\alpha_y^l}\dot{\alpha}_x^l+C_{\alpha_x^r}^2S_{\alpha_y^r}\dot{\alpha}_x^r\big)+\eta_4\big(C_{\alpha_x^r}S_{\alpha_x^r}-C_{\alpha_x^l}S_{\alpha_x^l}\big)S_{\phi_x}\dot{\phi}_x\\
&+\gamma_2C_{\phi_x}S_{\phi_y}\big(C_{\alpha_x^l}\dot{\alpha}_x^l+C_{\alpha_x^r}\dot{\alpha}_x^r\big)+\gamma_2\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big)C_{\phi_x}C_{\phi_y}\dot{\phi}_y\\
&+\eta_4S_{2\phi_x}\big(S_{\alpha_x^l}-S_{\alpha_x^r}\big)\dot{\phi}_y+\eta_2\big(C_{\alpha_x^l}^2S_{\alpha_y^l}C_{\alpha_y^l}+C_{\alpha_x^r}^2S_{\alpha_y^r}C_{\alpha_y^r}\big)C_{\phi_x}\dot{\phi}_x\\
&-\eta_2\big(C_{\alpha_x^l}S_{\alpha_x^l}C_{\alpha_y^l}+C_{\alpha_x^r}^2S_{\alpha_x^r}C_{\alpha_y^r}\big)C_{\phi_x}^2\dot{\phi}_y-\gamma_2S_{\phi_x}S_{\phi_y}\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big)\dot{\phi}_x\\
&+2\eta_4\big(C_{\alpha_x^r}C_{\alpha_y^r}-C_{\alpha_x^l}C_{\alpha_y^l}\big)C_{\phi_x}^2\dot{\phi}_y-\eta_2\big(C_{\alpha_x^l}^2C_{\alpha_y^l}^2+C_{\alpha_x^r}^2C_{\alpha_y^r}^2\big)S_{\phi_x}C_{\phi_x}\dot{\phi}_y\\
&+\eta_3\big(C_{\alpha_x^l}C_{\alpha_y^l}+C_{\alpha_x^r}C_{\alpha_y^r}\big)S_{2\phi_x}\dot{\phi}_y+\gamma_2C_{\alpha_x^l}C_{\alpha_y^l}\big(C_{\phi_x}S_{\phi_y}\dot{\phi}_x+S_{\phi_x}C_{\phi_y}\dot{\phi}_y\big)\\
&+\gamma_2C_{\alpha_x^r}C_{\alpha_y^r}\big(C_{\phi_x}S_{\phi_y}\dot{\phi}_x+S_{\phi_x}C_{\phi_y}\dot{\phi}_y\big)-\gamma_2S_{\phi_x}S_{\phi_y}\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l+C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big)\\
&-\gamma_2S_{\phi_x}S_{\phi_y}\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r+C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big)-\eta_2S_{\phi_x}C_{\alpha_x^l}C_{\alpha_y^l}S_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_x^l\\
&-\eta_2S_{\phi_x}C_{\alpha_x^r}C_{\alpha_y^r}S_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_x^r
\end{aligned}
\tag{A.30}
$$

$$
\begin{aligned}
C_{\phi\phi}^{22}(q,\dot{q}) \;=\;& \big(\eta_6-\eta_5+\eta_2-\frac{1}{2}\eta_1\big)S_{2\phi_x}\dot{\phi}_x+\eta_4\big(C_{\alpha_x^l}\dot{\alpha}_x^l-C_{\alpha_x^r}\dot{\alpha}_x^r\big)-\eta_3\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big)\dot{\phi}_x\\
&+\eta_2\big(S_{\alpha_x^l}C_{\alpha_x^l}C_{\alpha_x^r}+S_{\alpha_x^r}C_{\alpha_x^r}C_{\alpha_x^r}\big)\dot{\phi}_x+\frac{1}{2}\eta_2S_{2\phi_x}\big(C_{\alpha_y^l}\dot{\alpha}_x^l+C_{\alpha_y^r}\dot{\alpha}_x^r\big)\\
&+\eta_4\big(C_{\alpha_x^l}C_{\alpha_y^l}-C_{\alpha_x^r}C_{\alpha_y^r}\big)\dot{\phi}_x+\eta_2C_{\alpha_x^l}C_{\alpha_y^l}\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l+C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big)\\
&+\eta_2C_{\alpha_x^r}C_{\alpha_y^r}\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r+C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big)-\big(\gamma_1+2\gamma_3\big)\big(S_{\phi_x}C_{\phi_y}\dot{\phi}_x+C_{\phi_x}S_{\phi_y}\dot{\phi}_y\big)\\
&-\eta_2C_{\phi_x}^2\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l+S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r\big)-\frac{1}{2}\eta_2\big(C_{\alpha_x^l}^2+C_{\alpha_x^r}^2\big)S_{2\phi_x}\dot{\phi}_x\\
&-\eta_4C_{\phi_x}^2\big(C_{\alpha_x^l}\dot{\alpha}_x^l-C_{\alpha_x^r}\dot{\alpha}_x^r\big)+2\eta_3\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big)C_{\phi_x}^2\dot{\phi}_x\\
&+\gamma_2S_{\phi_y}\big(C_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_y^l+C_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_y^r\big)+\gamma_2\big(C_{\alpha_x^l}S_{\alpha_y^l}+C_{\alpha_x^r}S_{\alpha_y^r}\big)C_{\phi_y}\dot{\phi}_y\\
&+\frac{1}{2}\eta_3S_{2\phi_x}\big(C_{\alpha_x^l}\dot{\alpha}_x^l+C_{\alpha_x^r}\dot{\alpha}_x^r\big)+\gamma_2S_{\phi_x}C_{\phi_y}\big(C_{\alpha_x^l}\dot{\alpha}_x^l+C_{\alpha_x^r}\dot{\alpha}_x^r\big)\\
&+\gamma_2\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big)C_{\phi_x}C_{\phi_y}\dot{\phi}_x+\eta_4\big(S_{\alpha_x^l}-S_{\alpha_x^r}\big)S_{2\phi_x}\dot{\phi}_x\\
&-2\eta_2\big(S_{\alpha_x^l}C_{\alpha_x^l}C_{\alpha_y^l}+S_{\alpha_x^r}C_{\alpha_x^r}C_{\alpha_y^r}\big)C_{\phi_x}^2\dot{\phi}_x-\gamma_2S_{\phi_y}\big(S_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_x^l+S_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_x^r\big)\\
&-\eta_2S_{2\phi_x}\big(C_{\alpha_x^l}^2C_{\alpha_y^l}\dot{\alpha}_x^l+C_{\alpha_x^r}^2C_{\alpha_y^r}\dot{\alpha}_x^r-S_{\alpha_x^r}\big)-\gamma_2\big(S_{\alpha_x^l}+S_{\alpha_x^r}\big)S_{\phi_x}S_{\phi_y}\dot{\phi}_y\\
&-2\eta_4\big(C_{\alpha_x^l}C_{\alpha_y^l}-C_{\alpha_x^r}C_{\alpha_y^r}\big)C_{\phi_x}^2\dot{\phi}_x+\eta_3C_{\phi_x}^2\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l+C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big)\\
&+\eta_3C_{\phi_x}^2\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r+C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big)-\eta_2C_{\phi_x}^2C_{\alpha_x^l}C_{\alpha_y^l}\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l+C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big)\\
&-\eta_2C_{\phi_x}^2C_{\alpha_x^r}C_{\alpha_y^r}\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r+C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big)-\frac{1}{2}\eta_2\big(C_{\alpha_x^l}^2C_{\alpha_y^l}^2+C_{\alpha_x^r}^2C_{\alpha_y^r}^2\big)S_{2\phi_x}\dot{\phi}_x
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{1}{2}\eta_2 S_{2\phi_x}\big(S_{\alpha_x^l}C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l + S_{\alpha_x^r}C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big) + \eta_3\big(C_{\alpha_x^l}C_{\alpha_y^l} + C_{\alpha_x^r}C_{\alpha_y^r}\big)S_{2\phi_x}\dot{\phi}_x \\
&+\gamma_2 C_{\phi_x}C_{\phi_y}\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l + C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big) + \gamma_2 C_{\phi_x}C_{\phi_y}\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r + C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big) \\
&+\frac{1}{2}\eta_4 S_{2\phi_x}\big(S_{\alpha_x^l}C_{\alpha_y^l}\dot{\alpha}_x^l + C_{\alpha_x^l}S_{\alpha_y^l}\dot{\alpha}_y^l\big) - \frac{1}{2}\eta_4 S_{2\phi_x}\big(S_{\alpha_x^r}C_{\alpha_y^r}\dot{\alpha}_x^r + C_{\alpha_x^r}S_{\alpha_y^r}\dot{\alpha}_y^r\big) \\
&+\gamma_2\big(C_{\alpha_x^l}C_{\alpha_y^l} + C_{\alpha_x^r}C_{\alpha_y^r}\big)\big(S_{\phi_x}C_{\phi_y}\dot{\phi}_x + C_{\phi_x}S_{\phi_y}\dot{\phi}_y\big)
\end{aligned} \tag{A.31}
$$

The vector of gravitational forces $G(q)$ in Eq. A.1 is given by:

$$
G(q) = \begin{bmatrix} \mathbf{0} \\ G_{\alpha_l}(q_s) \\ G_{\alpha_r}(q_s) \\ G_{\phi}(q_s) \end{bmatrix}, \tag{A.32}
$$

where each $G_{ij} \in \mathbb{R}^{2\times 1}$ is given below.

$$
G_{\alpha^l}(q) = \begin{bmatrix} \chi_2\big(S_{\phi_x}C_{\phi_y}C_{\alpha_x^l} - S_{\phi_y}S_{\alpha_x^l}S_{\alpha_y^l} + C_{\phi_x}C_{\phi_y}S_{\alpha_x^l}C_{\alpha_y^l}\big) \\ \hdashline \\ \chi_2 C_{\alpha_x^l}\big(S_{\phi_y}C_{\alpha_y^l} + C_{\phi_x}C_{\phi_y}S_{\alpha_y^l}\big) \end{bmatrix}, \tag{A.33}
$$

$$
G_{\alpha^r}(q) = \begin{bmatrix} \chi_2\big(S_{\phi_x}C_{\phi_y}C_{\alpha_x^r} - S_{\phi_y}S_{\alpha_x^r}S_{\alpha_y^r} + C_{\phi_x}C_{\phi_y}S_{\alpha_x^r}C_{\alpha_y^r}\big) \\ \hdashline \\ \chi_2 C_{\alpha_x^r}\big(S_{\phi_y}C_{\alpha_y^r} + C_{\phi_x}C_{\phi_y}S_{\alpha_y^r}\big) \end{bmatrix}, \tag{A.34}
$$

$$
G_{\phi}(q) = \begin{bmatrix} C_{\phi_y}\Big(\chi_2\big(C_{\phi_x}\big(S_{\alpha_x^l} + S_{\alpha_x^r}\big) + S_{\phi_x}\big(C_{\alpha_x^l}C_{\alpha_y^l} + C_{\alpha_x^r}C_{\alpha_y^r}\big)\big)\Big) - \big(\chi_1 + 2\chi_3\big)S_{\phi_x}C_{\phi_y} \\ \hdashline \\ \chi_2\Big(C_{\phi_y}\big(C_{\alpha_x^l}S_{\alpha_y^l} + C_{\alpha_x^r}S_{\alpha_y^r}\big) + S_{\phi_y}\big(C_{\phi_x}\big(C_{\alpha_x^l}C_{\alpha_y^l} + C_{\alpha_x^r}C_{\alpha_y^r}\big) - S_{\phi_x}\big(S_{\alpha_x^l} + S_{\alpha_x^r}\big)\big)\Big) \\ -\big(\chi_1 + 2\chi_3\big)C_{\phi_x}S_{\phi_y} \end{bmatrix}. \tag{A.35}
$$

Appendix B

Verification of properties for shape-accelerated balancing systems

Section 4.1.2 presented a list of properties that shape-accelerated balancing systems satisfy, and some of these properties are exploited in the design of the shape trajectory planner presented in Sec. 4.2. The ballbot both with and without arms is a shape-accelerated balancing system, and this appendix verifies that the dynamics of the ballbot satisfy the properties of shape-accelerated balancing systems listed in Sec. 4.1.2.

B.1 The 3D ballbot without arms

The dynamics of the 3D ballbot model without arms presented in Sec. 3.3.1 is verified to satisfy all properties of shape-accelerated balancing systems listed in Sec. 4.1.2 as follows.

(a) The ball angles $\theta = [\theta_x, \theta_y]^T \in \mathbb{R}^{2\times 1}$ form the position variables q_x, while the body angles $\phi = [\phi_x, \phi_y]^T \in \mathbb{R}^{2\times 1}$ form the shape variables q_s. All position variables are actuated, while all the shape variables are unactuated, $i.e.$, $n_{s_a} = 0$. Therefore, the number of actuated variables equals the number of position variables.

(b) The ballbot without arms has one unactuated shape set, $i.e.$ the body angles $\phi \in \mathbb{R}^{2\times 1}$, and no actuated shape sets.

(c) The vector of gravitational forces shown earlier in Eq. 3.5 is given by

$$G(q) \;=\; \begin{bmatrix} 0 \\ 0 \\ -m_b g \ell_b \sin(\phi_x)\cos(\phi_y) \\ -m_b g \ell_b \cos(\phi_x)\sin(\phi_y) \end{bmatrix} \in \mathbb{R}^{4\times 1}, \tag{B.1}$$

where m_b is the mass of the body, ℓ_b is the height of the body center of mass from the ball center, and g is the acceleration due to gravity. These system parameters can be found in Table 3.1.

It can be seen from Eq. B.1 that the vector of gravitational forces $G(q)$ is a function of only the shape variables q_s and is independent of the position variables q_x.

(d) The vector of Coriolis and centrifugal forces can be obtained from Eq. 3.4 as follows:

$$C(q,\dot{q})\dot{q} \;=\; \begin{bmatrix} -\gamma_2\Big(C_{\phi_x}S_{\phi_y}\big(\dot{\phi}_x^2+\dot{\phi}_y^2\big)+2S_{\phi_x}C_{\phi_y}\dot{\phi}_x\dot{\phi}_y\Big) \\ \gamma_2 S_{\phi_x}\dot{\phi}_x^2 \\ S_{\phi_x}\big(-\gamma_2\dot{\phi}_x^2+\gamma_3 C_{\phi_x}\dot{\phi}_y^2\big) \\ -\gamma_2 C_{\phi_x}S_{\phi_y}\big(\dot{\phi}_x^2+\dot{\phi}_y^2\big)-2S_{\phi_x}\dot{\phi}_x\dot{\phi}_y\big(\gamma_2 C_{\phi_y}+\gamma_3 C_{\phi_x}\big) \end{bmatrix} \in \mathbb{R}^{4\times 1}, \tag{B.2}$$

where $C_i = \cos(i)$, $S_i = \sin(i)$, $\gamma_2 = m_b\ell_b r$, and $\gamma_3 = m_b\ell^2 + I^b_{yy} - I^b_{zz}$.

It can be seen from Eq. B.2 that the vector of Coriolis and centrifugal forces $C(q,\dot{q})\dot{q}$ is independent of both the position and velocity of position variables, *i.e.*, q_x and $\dot{q}_x$.

(e) The system matrices obtained by the Jacobian linearization at the origin are given below.

$$A \;=\; \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \dfrac{-\xi_1}{\xi_3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \dfrac{\xi_1}{\xi_4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \dfrac{\xi_2}{\xi_4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{-\xi_2}{\xi_3} & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{8\times 8}, \tag{B.3}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{-\xi_5}{\xi_3} & 0 \\ 0 & \frac{-\xi_6}{\xi_4} \\ 0 & \frac{-\xi_7}{\xi_4} \\ \frac{\xi_7}{\xi_3} & 0 \end{bmatrix} \in \mathbb{R}^{8\times 2}, \tag{B.4}$$

where, $\xi_1 = m_b g \ell_b \big(I_w + m_b \ell_b r + (m_b + m_w) r^2\big)$, $\xi_2 = m_b g \ell_b \big(I_w + (m_b + m_w) r^2\big)$, $\xi_3 = I_w(I^b_{yy} + m_b \ell_b^2) + I^b_{yy} r^2 (m_b + m_w) + m_b m_w \ell_b^2 r^2$, $\xi_4 = I_w(I^b_{xx} + m_b \ell_b^2) + I^b_{xx} r^2 (m_b + m_w) + m_b m_w \ell_b^2 r^2$, $\xi_5 = I_w + I^b_{yy} + 2 m_b \ell_b r + (m_b + m_w) r^2 + m_b \ell_b^2$, $\xi_6 = I_w + I^b_{xx} + 2 m_b \ell_b r + (m_b + m_w) r^2 + m_b \ell_b^2$, and $\xi_7 = I_w + m_b \ell_b r + (m_b + m_w) r^2$.

The pair (A, B) formed by the linear system matrices shown in Eq. B.3 and Eq. B.4 is controllable if the controllability matrix P shown in Eq. B.5 has full row rank.

$$P = \begin{bmatrix} B & AB & A^2B & A^3B & A^4B & A^5B & A^6B & A^7B \end{bmatrix} \in \mathbb{R}^{8\times 16} \tag{B.5}$$

Since ξ_1 to ξ_7 are all positive and non-zero, it can be verified that the controllability matrix P has full row rank, $i.e.$, rank(P) = 8.

(f) The system is said to have unstable zero dynamics [44] if the matrix given below is negative definite.

$$\frac{\partial\big(M^{-1}_{s_u s_u}(q_s) G_{s_u}(q_s)\big)}{\partial q_{s_u}}\bigg|_{q_s = \mathbf{0}} = \frac{-\rho_2}{\rho_1(\rho_1 + I^b_{xx} + I^b_{yy}) + I^b_{xx} I^b_{yy}} \begin{bmatrix} \rho_1 + I^b_{yy} & 0 \\ 0 & \rho_1 + I^b_{xx} \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.6}$$

where $\rho_1 = I_w + m_w r^2 + m_b(\ell_b + r)^2$ and $\rho_2 = m_b g \ell_b$.

It can be seen that the Jacobian linearization at the origin shown in Eq. B.6 is a function only of the system parameters and is always negative definite.

(g) The system is said to have locally strong inertial coupling [118] if the matrix given below is

invertible.

$$M_{s_u x}(q_s) \;=\; \begin{bmatrix} -\gamma_2 S_{\phi_x} S_{\phi_y} & -\gamma_1 - \gamma_2 C_{\phi_x} \\ \gamma_1 + \gamma_2 C_{\phi_x} C_{\phi_y} & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.7}$$

where $C_i = \cos(i)$, $S_i = \sin(i)$, $\gamma_1 = I_w + (m_b + m_w)r^2$ and $\gamma_2 = m_b \ell r$, and its determinant is given by:

$$\det(M_{s_u x}(q_s)) \;=\; (\gamma_1 + \gamma_2 C_{\phi_x})(\gamma_1 + \gamma_2 C_{\phi_x} C_{\phi_y}). \tag{B.8}$$

The matrix $M_{s_u x}(q_s)$ shown in Eq. B.7 loses its rank when its determinant shown in Eq. B.8 is zero. This happens when either $\cos\phi_x = -\dfrac{\gamma_1}{\gamma_2}$ or $\cos\phi_x \cos\phi_y = -\dfrac{\gamma_1}{\gamma_2}$. Since γ_1 and γ_2 are both positive, either $\cos\phi_x$ or $\cos\phi_x \cos\phi_y$ must be negative for the matrix $M_{s_u x}(q_s)$ to lose rank. But this will not happen for $\phi_x \in \left[-\dfrac{\pi}{2}, \dfrac{\pi}{2}\right]$ and $\phi_y \in \left[-\dfrac{\pi}{2}, \dfrac{\pi}{2}\right]$ irrespective of the values of γ_1 and γ_2. Therefore, $M_{s_u x}(q_s)^{-1}$ exists for the entire range of body angle values of interest. This ensures that the system has locally strong inertial coupling [118].

(h) The Jacobian linearization of the vector of gravitational forces corresponding to the unactuated shape variables $G_{s_u}(q_s)$ w.r.t. q_s at $q_s = \mathbf{0}$ is given by:

$$\left.\frac{\partial G_{s_u}(q_s)}{\partial q_s}\right|_{q_s=\mathbf{0}} \;=\; -m_b g \ell_b \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}. \tag{B.9}$$

It can be seen from Eq. B.9 that the Jacobian linearization of $G_{s_u}(q_s)$ w.r.t. q_s at the origin is a function of only the system parameters, and it always exists and is invertible.

(i) The Jacobian linearization of $M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)$ w.r.t. q_s at $q_s = \mathbf{0}$ is given by:

$$\left.\frac{\partial\left(M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)\right)}{\partial q_s}\right|_{q_s=\mathbf{0}} \;=\; \frac{m_b g \ell_b}{\gamma_1 + \gamma_2} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.10}$$

where $\gamma_1 = I_w + (m_b + m_w)r^2$ and $\gamma_2 = m_b \ell r$.
It can be seen from Eq. B.10 that the Jacobian linearization of $M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)$ w.r.t. q_s at the origin is a function of only the system parameters, and it always exists and is invertible.

B.2 The 3D ballbot with a pair of 2-DOF arms

The dynamics of the 3D ballbot model with a pair of 2-DOF arms presented in Appendix A is verified to satisfy all properties of shape-accelerated balancing systems listed in Sec. 4.1.2 as follows.

(a) The ball angles $\theta = [\theta_x, \theta_y]^T \in \mathbb{R}^{2\times 1}$ form the actuated position variables q_x, while the body angles $\phi = [\phi_x, \phi_y]^T \in \mathbb{R}^{2\times 1}$ form the unactuated shape variables q_{s_u}, and the arm angles $[\alpha^l, \alpha^r]^T = [\alpha^l_x, \alpha^l_y, \alpha^r_x, \alpha^r_y]^T \in \mathbb{R}^{4\times 1}$ form the actuated shape variables q_{s_a}. Therefore, number of unactuated variables equals the number of position variables.

(b) The ballbot with a pair of 2-DOF arms has one unactuated shape set, $i.e.$ the body angles $\phi \in \mathbb{R}^{2\times 1}$, and two actuated shape sets, one for each arm, $i.e.$, the left arm angles $\alpha^l \in \mathbb{R}^{2\times 1}$ and the right arm angles $\alpha^r \in \mathbb{R}^{2\times 1}$.

(c) The vector of gravitational forces for the ballbot with arms shown earlier in Eq. A.32 is given by

$$G(q) = \begin{bmatrix} \mathbf{0} \\ G_{\alpha_l}(q_s) \\ G_{\alpha_r}(q_s) \\ G_{\phi}(q_s) \end{bmatrix} \in \mathbb{R}^{8\times 1}, \tag{B.11}$$

where each $G_{ij}(q_s) \in \mathbb{R}^{2\times 1}$ can be found in Eq. A.33−A.35.
It can be seen from Eq. B.11 that the vector of gravitational forces $G(q)$ is a function of only the shape variables $q_s = [\alpha^l, \alpha^r, \phi]^T$, and is independent of the position variables q_x.

(d) The vector of Coriolis and centrifugal forces can be obtained from Eq. A.19 as follows:

$$C(q,\dot{q})\dot{q} = \begin{bmatrix} C_{\theta\alpha^l}(q_s,\dot{q}_s)\dot{\alpha}^l + C_{\theta\alpha^r}(q_s,\dot{q}_s)\dot{\alpha}^r + C_{\theta\phi}(q_s,\dot{q}_s)\dot{\phi} & \in \mathbb{R}^{2\times 1} \\ C_{\alpha^l\alpha^l}(q_s,\dot{q}_s)\dot{\alpha}^l + C_{\alpha^l\phi}(q_s,\dot{q}_s)\dot{\phi} & \in \mathbb{R}^{2\times 1} \\ C_{\alpha^r\alpha^r}(q_s,\dot{q}_s)\dot{\alpha}^r + C_{\alpha^r\phi}(q_s,\dot{q}_s)\dot{\phi} & \in \mathbb{R}^{2\times 1} \\ C_{\phi\alpha^l}(q_s,\dot{q}_s)\dot{\alpha}^l + C_{\phi\alpha^r}(q_s,\dot{q}_s)\dot{\alpha}^r + C_{\phi\phi}(q_s,\dot{q}_s)\dot{\phi} & \in \mathbb{R}^{2\times 1} \end{bmatrix} \in \mathbb{R}^{8\times 1}, \tag{B.12}$$

where each $C_{ij}(q_s, \dot{q}_s) \in \mathbb{R}^{2\times 2}$ can be found in Eq. A.20−A.29.
It can be seen from Eq. B.12 that the vector of Coriolis and centrifugal forces $C(q,\dot{q})\dot{q}$ is independent of both the position and velocity of position variables, $i.e.$, q_x and $\dot{q}_x$.

(e) The linear system matrices obtained by the Jacobian linearization of the nonlinear model at

the origin are given below.

$$A = \left[\begin{array}{cccccccc|c} & & & & \mathbf{0}_{8\times 8} & & & & \mathbf{I}_{8\times 8} \\ \hline 0 & 0 & 0 & \frac{\varrho_{11}}{\varrho_7} & 0 & \frac{\varrho_{11}}{\varrho_7} & 0 & -\frac{\varrho_1}{\varrho_7} & \\ 0 & 0 & -\frac{\varrho_{12}}{\varrho_8} & 0 & -\frac{\varrho_{12}}{\varrho_8} & 0 & \frac{\varrho_2}{\varrho_8} & 0 & \\ 0 & 0 & -\frac{\varrho_{13}}{\varrho_8\varrho_{10}} & 0 & \frac{\varrho_{14}}{\varrho_8\varrho_{10}} & 0 & -\frac{\varrho_3}{\varrho_8} & 0 & \\ 0 & 0 & 0 & -\frac{\varrho_{15}}{\varrho_7\varrho_9} & 0 & \frac{\varrho_{16}}{\varrho_7\varrho_9} & 0 & -\frac{\varrho_4}{\varrho_7} & \mathbf{0}_{8\times 8} \\ 0 & 0 & \frac{\varrho_{14}}{\varrho_8\varrho_{10}} & 0 & -\frac{\varrho_{13}}{\varrho_8\varrho_{10}} & 0 & -\frac{\varrho_3}{\varrho_8} & 0 & \\ 0 & 0 & 0 & \frac{\varrho_{16}}{\varrho_7\varrho_9} & 0 & -\frac{\varrho_{15}}{\varrho_7\varrho_9} & 0 & -\frac{\varrho_4}{\varrho_7} & \\ 0 & 0 & -\frac{\varrho_{17}}{\varrho_8} & 0 & -\frac{\varrho_{17}}{\varrho_8} & 0 & \frac{\varrho_5}{\varrho_8} & 0 & \\ 0 & 0 & 0 & -\frac{\varrho_{18}}{\varrho_7} & 0 & -\frac{\varrho_{18}}{\varrho_7} & 0 & \frac{\varrho_6}{\varrho_7} & \end{array}\right] \in \mathbb{R}^{16\times 16}, \tag{B.13}$$

$$B = \left[\begin{array}{cccccc} & & \mathbf{0}_{8\times 6} & & & \\ \hline \frac{\varrho_{19}}{\varrho_7} & 0 & 0 & \frac{\varrho_{21}}{\varrho_7} & 0 & \frac{\varrho_{21}}{\varrho_7} \\ 0 & \frac{\varrho_{20}}{\varrho_8} & \frac{\varrho_{22}}{\varrho_8} & 0 & \frac{\varrho_{22}}{\varrho_8} & 0 \\ 0 & \frac{\varrho_{22}}{\varrho_8} & \frac{\varrho_{25}}{\varrho_8} & 0 & \frac{\varrho_{26}}{\varrho_8\varrho_{10}} & 0 \\ \frac{\varrho_{21}}{\varrho_7} & 0 & 0 & \frac{\varrho_{30}}{\varrho_7\varrho_9} & 0 & \frac{\varrho_{28}}{\varrho_7\varrho_9} \\ 0 & \frac{\varrho_{22}}{\varrho_8} & \frac{\varrho_{26}}{\varrho_8\varrho_{10}} & 0 & \frac{\varrho_{25}}{\varrho_8} & 0 \\ \frac{\varrho_{21}}{\varrho_7} & 0 & 0 & \frac{\varrho_{28}}{\varrho_7\varrho_9} & 0 & \frac{\varrho_{30}}{\varrho_7\varrho_9} \\ 0 & \frac{\varrho_{23}}{\varrho_8} & -\frac{\varrho_{27}}{\varrho_8} & 0 & -\frac{\varrho_{27}}{\varrho_8} & 0 \\ -\frac{\varrho_{24}}{\varrho_7} & 0 & 0 & \frac{\varrho_{29}}{\varrho_7} & 0 & \frac{\varrho_{29}}{\varrho_7} \end{array}\right] \in \mathbb{R}^{16\times 6}, \tag{B.14}$$

where ϱ_1 to ϱ_{30} are non-zero, positive functions of system parameters.

The pair (A, B) formed by the linear system matrices shown in Eq. B.13 and Eq. B.14 is

controllable if the controllability matrix P shown in Eq. B.15 has full row rank.

$$P = \begin{bmatrix} B & AB & A^2B & A^3B & A^4B & \cdots & A^{14}B & A^{15}B \end{bmatrix} \in \mathbb{R}^{16\times 96} \tag{B.15}$$

Since ϱ_1 to ϱ_{30} are all positive and non-zero, it can be verified that the controllability matrix P has full row rank, *i.e.*, rank(P) = 16.

(f) The system is said to have unstable zero dynamics [44] if the matrix given below is negative definite.

$$\frac{\partial\big(M^{-1}_{s_u s_u}(q_s)G_{s_u}(q_s)\big)}{\partial q_{s_u}}\bigg|_{q_s=\mathbf{0}} = \frac{-\rho_4}{\rho_5}\begin{bmatrix} \rho_3+I^b_{yy} & 0 \\ 0 & \rho_3+2\eta_6+I^b_{xx} \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.16}$$

where $\eta_6 = m_a d_a^{y2}$, $\rho_3 = I_w + m_w r^2 + m_b(\ell_b + r)^2 + 2m_a(d_a^z - \ell_a + r)^2$, $\rho_4 = m_b g \ell_b + 2m_a g(d_a^z - \ell_a)$, and $\rho_5 = \rho_3(\rho_3+2\eta_6+I^b_{xx}+I^b_{yy})+I^b_{yy}(I^b_{xx}+2\eta_6)$.
It can be seen that the Jacobian linearization at the origin shown in Eq. B.16 is a function only of the system parameters and their values are such that the Jacobian linearization at the origin is always negative definite.

(g) The system is said to have locally strong inertial coupling [118] if the matrix $M_{s_u x}(q_s) \in \mathbb{R}^{2\times 2}$ shown in A.15 is invertible. Its determinant is given by:

$$\det(M_{s_u x}(q_s)) = \Big(\beta + \varphi\Big)\Big(\beta + \varphi C_{\phi_y} + \gamma_2 S_{\phi_y}\big(C_{\alpha^l_x} S_{\alpha^l_y} + C_{\alpha^r_x} S_{\alpha^r_y}\big)\Big), \tag{B.17}$$

where $C_i = \cos(i)$, $S_i = \sin(i)$, $\beta = I_w + \big(m_w + m_b + 2m_a\big)r^2$, $\gamma_1 = m_b \ell_b r$, $\gamma_2 = m_a \ell_a r$, $\gamma_3 = m_a d_a^z r$, and $\varphi = S_{\phi_x}\big(\gamma_2(S_{\alpha^l_x} + S_{\alpha^r_x})\big) + C_{\phi_x}\big(\gamma_1 + 2\gamma_3 - \gamma_2(C_{\alpha^l_x} C_{\alpha^l_y} + C_{\alpha^r_x} C_{\alpha^r_y})\big)$.
It can be numerically shown that the $\det(M_{s_u x}(q_s)) \neq 0, \forall q_s \in \left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$. Therefore, $M_{s_u x}(q_s)^{-1}$ exists in a large neighborhood around the origin and hence the system has locally strong inertial coupling [118].

(h) The Jacobian linearization of the vector of gravitational forces corresponding to the unactuated shape variables $G_{s_u}(q_s)$ w.r.t. q_s at $q_s = \mathbf{0}$ is given by:

$$\frac{\partial G_{s_u}(q_s)}{\partial q_s}\bigg|_{q_s=\mathbf{0}} = \left[\frac{\partial G_{s_u}(q_s)}{\partial \alpha^l}\bigg|_{q_s=\mathbf{0}}, \frac{\partial G_{s_u}(q_s)}{\partial \alpha^r}\bigg|_{q_s=\mathbf{0}}, \frac{\partial G_{s_u}(q_s)}{\partial \phi}\bigg|_{q_s=\mathbf{0}}\right] \in \mathbb{R}^{2\times 6}, \tag{B.18}$$

$$\left.\frac{\partial G_{s_u}(q_s)}{\partial \alpha^l}\right|_{q_s=\mathbf{0}} = \chi_2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.19}$$

$$\left.\frac{\partial G_{s_u}(q_s)}{\partial \alpha^r}\right|_{q_s=\mathbf{0}} = \chi_2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.20}$$

$$\left.\frac{\partial G_{s_u}(q_s)}{\partial \phi}\right|_{q_s=\mathbf{0}} = -(\chi_1 - 2\chi_2 + 2\chi_3) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.21}$$

where $\chi_1 = m_b g \ell_b$, $\chi_2 = m_a g \ell_a$, and $\chi_3 = m_a g d_a^z$.
It can be seen from Eq. B.18 that the Jacobian linearization of $G_{s_u}(q_s)$ w.r.t. q_s at the origin is a function of only the system parameters, and it always exists but is not invertible. However, the Jacobian linearization of $G_{s_u}(q_s)$ w.r.t. every single shape set at the origin shown in Eq. B.19−B.21 exists and is invertible.

(i) The Jacobian linearization of $M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)$ w.r.t. every shape set at $q_s = \mathbf{0}$ are given below.

$$\left.\frac{\partial\big(M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)\big)}{\partial \alpha^l}\right|_{q_s=\mathbf{0}} = \frac{\chi_2}{\beta + \gamma_1 - 2\gamma_2 + 2\gamma_3} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.22}$$

$$\left.\frac{\partial\big(M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)\big)}{\partial \alpha^r}\right|_{q_s=\mathbf{0}} = \frac{\chi_2}{\beta + \gamma_1 - 2\gamma_2 + 2\gamma_3} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.23}$$

$$\left.\frac{\partial\big(M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)\big)}{\partial \phi}\right|_{q_s=\mathbf{0}} = \frac{\chi_1 - 2\chi_2 + 2\chi_3}{\beta + \gamma_1 - 2\gamma_2 + 2\gamma_3} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{B.24}$$

where $\gamma_1 = m_b \ell_b r$, $\gamma_2 = m_a \ell_a r$, $\gamma_3 = m_a d_a^z r$, $\chi_1 = m_b g \ell_b$, $\chi_2 = m_a g \ell_a$, and $\chi_3 = m_a g d_a^z$.
The Jacobian linearization of $M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)$ w.r.t. q_s at the origin is a function of only the system parameters, and it always exists but is not invertible. However, the Jacobian linearization of $M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)$ w.r.t. every single shape set at the origin shown in Eq. B.22−B.24 exists and is invertible.

Appendix C

Software architecture

One of the major contributions of the work presented in this thesis is the development of the software architecture that enabled the ballbot to achieve all the experimental results presented in this thesis. This appendix presents a brief description of the ballbot's software architecture.

The ballbot has two single-board computers on its decks: (i) a single-core computer running QNX real-time operating system, and (ii) a dual-core computer running Ubuntu 10.04 LTS operating system. The QNX computer performs low-level control operations like balancing control, arm control, yaw control and leg control. These control operations include reading low-level sensors like IMU, absolute yaw encoder, ball encoders, arm encoders and leg encoders, and also include providing motor commands to ball, arm, yaw and leg motors. The Linux computer performs high-level operations like localization and motion planning. These operations include reading the laser scanner data to update an occupany grid map of the environment. The QNX and Linux computers communicate with each other using sockets via a wired connection.

The ballbot's software architecture includes a graphical user interface (GUI), which allows the user to monitor and control the ballbot. This graphical user interface is run on a laptop, which wirelessly communicates with the ballbot's Linux computer using sockets. The graphical user interface also includes a joystick controller that allows the user to control the ballbot using a joystick. A high-level overview of the ballbot's software architecture is shown in Fig. C.1.

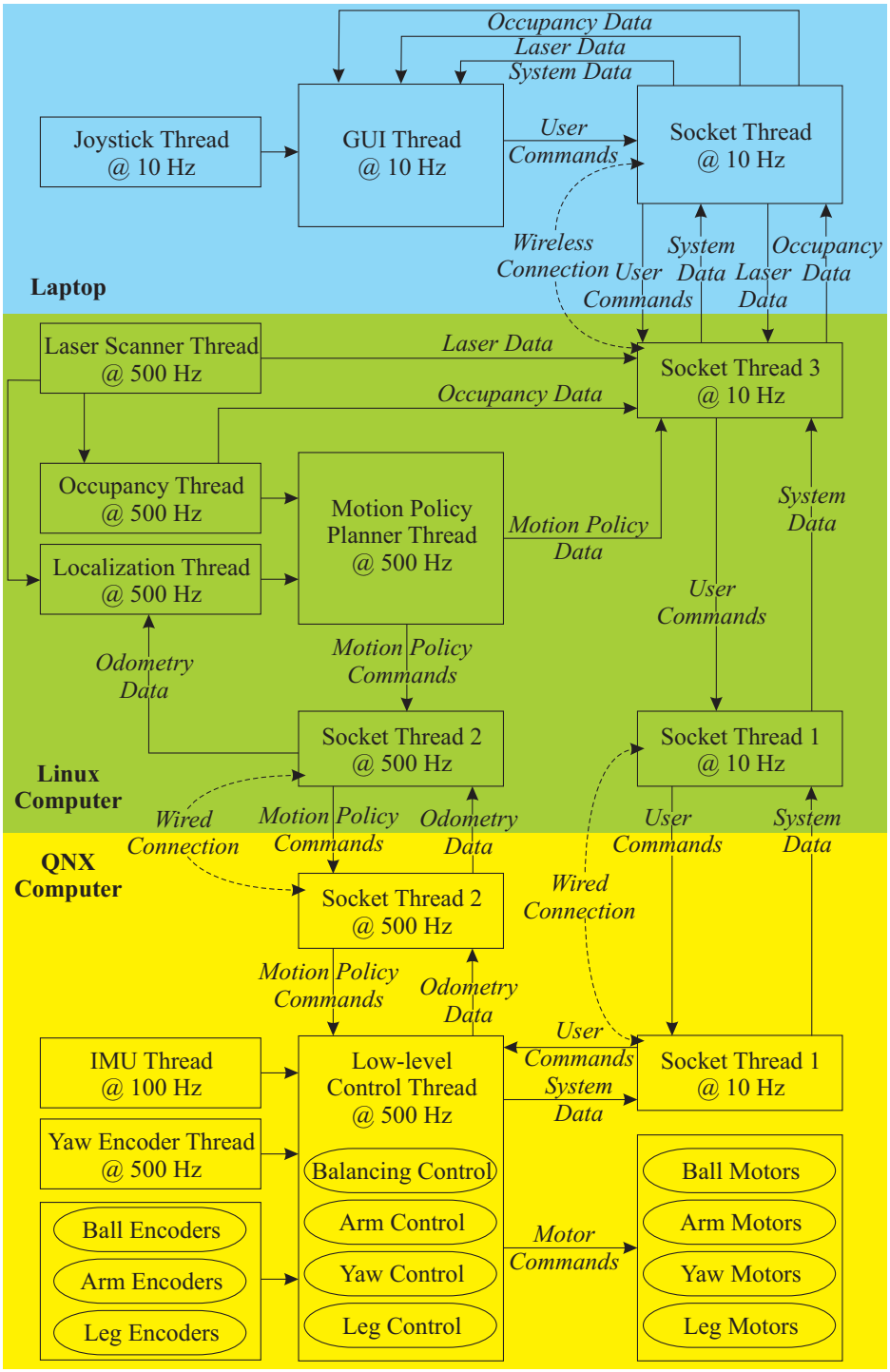


Figure C.1: A high-level overview of the ballbot's software architecture.

Appendix D

Links to the ballbot videos

Chapters 3–5 presented several successful experimental results on the ballbot, and links to the videos of the ballbot achieving these results are listed below.

D.1. Introduction to the ballbot:
http://www.youtube.com/watch?v=39zeZwlVaN0

D.2. Human-robot physical interaction with the ballbot:
http://www.youtube.com/watch?v=miZ_ebjoiFY

D.3. Fast motions for the ballbot without arms:
http://www.youtube.com/watch?v=Wd03f_6utA8

D.4. Planning in high-dimensional shape space:
http://www.youtube.com/watch?v=YX3DuIA9FLo

D.5. Planning with additional arm constraints:
http://www.youtube.com/watch?v=BW78KmmB9IU

D.6. Fast maneuvers for the ballbot without arms:
http://www.youtube.com/watch?v=VWAF3jTZLgw

D.7. Graceful navigation to achieve point-point and surveillance tasks:
http://www.youtube.com/watch?v=tzNKiGq5oaA

D.8. Surveillance motion with ten goals, and point-point motion with dynamic obstacles:
http://www.youtube.com/watch?v=tLDjZDRA1uE

D.9. Graceful navigation using regions:
http://www.youtube.com/watch?v=Ss7gsC3rQkk

Bibliography

[1] Pierre-Antoine Absil and Rodolphe Sepulchre. A hybrid control scheme for swing-up acrobatics. In *Proceedings of the 5th European Control Conference*, pages 2860–2864, 2001. 13

[2] F. Amirabdollahian, R. Loureiro, and W. Harwin. Minimum jerk trajectory control for rehabilitation and haptic applications. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3380–3385, 2002. 6

[3] Anybots. http://anybots.com. 11

[4] G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Magaritis, M. Montemerlo, J. Pineau, N. Roy, J. Schulte, and S. Thrun. Towards personal service robots for the elderly. In *Proc. Workshop on Interactive Robotics and Entertainment (WIRE)*, Pittsburgh, PA, 2000. xiii, 2

[5] C. Belta, V. Iser, and G. J. Pappas. Discrete abstractions for robot planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21(5):864–874, 2005. 15

[6] J. Biswas, B. Coltin, and M. Veloso. Corrective gradient refinement for mobile robot localization. In *Intelligent Robots and Systems (IROS), 2011 IEEE International Conference on*. IEEE, 2011. 99, 100

[7] Anthony M. Bloch, P. S. Krishnaprasad, Jerrold E. Marsden, and Richard M. Murray. Nonholonomic mechanical systems with symmetry. *Arch. Rational Mech. Anal.*, 136: 21–99, 1996. 13

[8] R. W. Brockett. Formal languages for motion description and map making. *Robotics, RI: Amer. Math. Soc.*, 41:181–193, 1990. 15

[9] F. Bullo and A. D. Lewis. Kinematic controllability and motion planning for the snakeboard. *IEEE Transactions on Robotics and Automation*, 19:494–498, 2003. 13

[10] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, 1999. xvii, 14, 76, 77, 78, 85, 109, 113

[11] D. C. Conner, Howie Choset, and A. A. Rizzi. Integrated planning and control for convex-bodied nonholonomic systems using local feedback. In *Proc. Robotics: Science and Systems II*, pages 57–64, 2006. 14, 76, 77, 78, 89, 109, 113

[12] P. Deegan, B. Thibodeau, and R. Grupen. Designing a self-stabilizing robot for dynamic mobile manipulation. *Robotics: Science and Systems - Workshop on Manipulation for Human Environments*, 2006. 11

[13] P. Deegan, B. Thibodeau, and R. Grupen. Designing a self-stabilizing robot for dynamic mobile manipulation. In *Robotics: Science and Systems - Workshop on Manipulation for Human Environments*, August 2006. 3

[14] P. Deegan, R. Grupen, A. Hanson, E. Horrell, S. Ou, E. Riseman, S. Sen, B. Thibodeau, A. Williams, and D. Xie. Mobile manipulators for assisted living in residential settings. *Autonomous Robots, Special Issue on Socially Assistive Robotics*, 24(2):179–192, 2008. 11

[15] S. Devasia and B. Paden. Exact output tracking for nonlinear time-varying systems. In *IEEE International Conference on Decision and Control*, volume 3, pages 2346–2355, 1994. 12

[16] S. Devasia, D. Chen, and B. Paden. Nonlinear inversion-based output tracking. In *IEEE Transactions on Automatic Control*, volume 41, pages 930–942, 1996. 12

[17] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. 89

[18] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 96, 99

[19] T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience*, 5:1688–1703, 1985. 6

[20] E. Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of Technology, June 2001. 16

[21] E. Frazzoli. Explicit solutions for optimal maneuver-based motion planning. In *Proc. IEEE Conference on Decision and Control*, volume 4, pages 3372–3377, 2003. 16

[22] E. Frazzoli, M. A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicle motion planning. In *Proc. IEEE Conference on Decision and Control*, pages 821–826, 2000. 16

[23] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA J. Guid., Control, Dynam.*, 25(1), 2002. 16

[24] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6), 2005. 16, 79, 82

[25] V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, and E. Feron. Aggressive maneuvering of small autonomous helicopters: A human-centered approach. *International Journal of Robotics Research*, 20(10), 2001. 16

[26] N. Getz. Control of balance for a nonlinear nonholonomic non-minimum phase model of a bicycle. In *American Control Conference, 1994*, volume 1, pages 148 – 151, 1994. 12

[27] Neil Getz and J. Karl Hedrick. An internal equilibrium manifold method of tracking for nonlinear nonminimum phase systems. In *in 1995 American Control Conference, (Seattle), American Automatic Control Council*, pages 2241–2245, 1995. 12

[28] Neil H. Getz. Tracking with balance. In *in 13th IFAC Triennial World Congress*, San Francisco, USA, 1996. 12

[29] Neil Holden Getz. *Dynamic Inversion of Nonlinear Maps with Applications to Nonlinear Control and Robotics*. PhD thesis, University of California at Berkeley, December 1995. 12, 47

[30] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–132, 2005. 59, 112

[31] J.W. Grizzle, Jonathan Hurst, Benjamin Morris, Hae-Won Park, and Koushil Sreenath. Mabel, a new robotic bipedal walker and runner. In *American Control Conference*, St. Louis, MO, June 2009. 3

[32] Shilpa Gulati and Benjamin Kuipers. High performance control for graceful motion of an intelligent wheelchair. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3932–3938, 2008. 6

[33] Y.-S. Ha and S. Yuta. Trajectory tracking control for navigation of self-contained mobile inverse pendulum. In *Proc. IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems*, pages 1875–1882, 1994. 11

[34] Y.S. Ha and S. Yuta. Indoor navigation of an inverse pendulum type autonomous mobile robot with adaptive stabilization control system. In *Experimental Robotics IV, Int'l. Symp.*, pages 529–37, 1997. 11

[35] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance*, 10(4):338–342, 1987. 58, 112

[36] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107, 1968. 89

[37] Ross Hatton and Howie Choset. Connection vector fields for underactuated systems. In *Proceedings of IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 451–456, 2008. 13

[38] L*á*szl*ó* Havasi. ERROSphere: an equilibrator robot. *Intl. Conf. on Control and Automation*, pages 971–976, June 27-29 2005. 11

[39] S. Hirose. *Biologically Inspired Robots: Snake-like Locomotors and Manipulators*. Oxford University Press, Oxford, 1993. 13

[40] N. Hogan. An organizing principle for a class of voluntary movements. *Journal of Neuroscience*, 4:2745–2754, 1984. 6

[41] Ralph Hollis. Ballbots. *Scientific American*, pages 72–78, Oct 2006. xiii, 2, 3, 17, 99

[42] T. M. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 2399–2404, 2005. 12

[43] iBot. http://www.ibotnow.com. 11

[44] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 1989. 12, 47, 137, 141

[45] A. Isidori and C. I. Byrnes. Output regulation of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(2):131–140, 1990. 12

[46] G. Kantor and A. A. Rizzi. Feedback control of underactuated systems via sequential composition: Visually guided control of a unicycle. In *11th International Symposium of Robotics Research*, Siena, Italy, October 2003. 14

[47] R. Kelly, J. Llamas, and R. Campa. A measurement procedure for viscous and coulomb friction. *IEEE Transactions on Instrumentation and Measurement*, 49(4):857–861, 2000.

28

[48] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal. Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation. In *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 546–553, Osaka, 1996. xiii, 2

[49] Jongwoo Kim and J.P. Ostrowski. Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints. *IEEE International Conference on Robotics and Automation*, 2:2200–2205, Sept. 2003. 12

[50] E. Klavins and D. E. Koditschek. A formalism for the composition of concurrent robot behaviors. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, volume 4, pages 3395–3402, 2000. 14

[51] Ross A. Knepper and Matthew T. Mason. Empirical sampling of path sets for local area motion planning. In *International Symposium on Experimental Robotics*, 2008. 80

[52] Ross A. Knepper and Matthew T. Mason. Path diversity is only part of the problem. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3260–3265, 2009. 80

[53] S. Koenig and M. Likachev. D* lite. In *Proc. 18th National Conf. on Artificial Intelligence*, pages 476–483, 2002. 115

[54] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Trans. Graphics (Special Issue: Proc. ACM SIGGRAPH 2002)*, 21(3):473–482, 2002. 15, 16

[55] Masaaki Kumagai and Takaya Ochiai. Development of a robot balancing on a ball. *Intl. Conf. on Control, Automation and Systems*, 2008. 11

[56] Masaaki Kumagai and Takaya Ochiai. Development of a robot balancing on a ball - application of passive motion to transport. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, pages 4106–4111, 2009. 11

[57] K. J. Kyriakopoulos and G. N. Saridis. Minimum jerk path generation. In *Proc. IEEE International Conference on Robotics and Automation*, pages 364–369, 1988. 6

[58] G. Lafferriere and H. Sussmann. Motion planning for controllable systems without drift. In *IEEE International Conference on Robotics and Automation*, pages 1148–1153, 1991. 12

[59] J. P. Laumond. *Robot Motion Planning and Control*. Springer-Verlag New York, Inc.,

Secaucus, NJ, USA, 1998. 12

[60] T. B. Lauwers, G. A. Kantor, and R. L. Hollis. A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Proc. Int'l. Conf. on Robotics and Automation*, Orlando, FL, May 15-19 2006. xiii, 2, 17, 19

[61] Tom Lauwers, George Kantor, and Ralph Hollis. One is enough! In *Proc. Int'l. Symp. for Robotics Research*, San Francisco, October 12-15 2005. Int'l. Foundation for Robotics Research. 6, 11, 17

[62] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. In *TR 98-11, Computer Science Dept., Iowa State University*, Oct. 1998. 16

[63] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2:164–168, 1944. 55, 59

[64] Andrew Lewis, Jim Ostrowski, Richard Murray, and Joel Burdick. Nonholonomic mechanics and locomotion: The snakeboard example. In *In Proc. IEEE Int. Conf. Robotics and Automation*, pages 2391–2397, 1994. 13

[65] A. Majumdar, M. Tobenkin, and R. Tedrake. Algebraic verification for parameterized motion planning libraries. In *Proc. American Control Conference*, 2012. 16

[66] A.K. Mampetta. Automatic transition of ballbot from statically stable state to dynamically stable state. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, 2006. CMU-RI-TR-01-00. 17, 35

[67] V. Manikonda, P. S. Krishnaprasad, and J. Hendler. A motion description language and a hybrid architecture for motion planning with nonholonomic robots. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, pages 2021–2028, 1995. 15

[68] V. Manikonda, P. S. Krishnaprasad, and J. Hendler. Languages, behaviors, hybrid architectures and motion control. *Mathematical Control Theory*, pages 199–226, 1998. 15

[69] A. Marigo and A. Bicchi. Steering driftless nonholonomic systems by control quanta. In *Proc. IEEE Conference on Decision and Control*, pages 4164–4169, 1998. 15, 16

[70] J.E. Marsden and M.J. Hoffman. *Elementary classical analysis*. W.H. Freeman, 1993. 50, 51, 54

[71] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Berkeley, 1994. 60

[72] Umashankar Nagarajan. Dynamic constraint-based optimal shape trajectory planner for shape-accelerated underactuated balancing systems. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, 2010. 45, 75, 78, 80, 112

[73] Umashankar Nagarajan and Ralph Hollis. Shape space planner for shape-accelerated balancing mobile robots. *International Journal of Robotics Research*, 2012. (Under Review). xiii, xiv, xv, xvi, xvii, xix, 4, 21, 25, 45, 56, 58, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 75, 112, 113, 119

[74] Umashankar Nagarajan, George Kantor, and Ralph Hollis. Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot. In *IEEE Int'l. Conf. on Robotics and Automation*, pages 3743–3748, 2009. xiv, 20, 22, 112

[75] Umashankar Nagarajan, George Kantor, and Ralph Hollis. Human-robot physical interaction with dynamically stable mobile robots. *4th ACM/IEEE Int'l. Conf. on Human-Robot Interaction*, March 11-13 2009. (Short paper and video). xiii, xv, 3, 4, 18, 38, 39, 40, 41

[76] Umashankar Nagarajan, Anish Mampetta, George Kantor, and Ralph Hollis. State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot. In *IEEE Int'l. Conf. on Robotics and Automation*, pages 998–1003, 2009. xiii, xiv, xv, 8, 19, 22, 27, 28, 29, 31, 32, 33, 34, 36, 37, 112

[77] Umashankar Nagarajan, George Kantor, and Ralph Hollis. Hybrid control for navigation of shape-accelerated underactuated balancing systems. In *Proc. IEEE Conference on Decision and Control*, pages 3566–3571, 2010. xiii, 4, 77, 78, 82, 83, 109, 113

[78] Umashankar Nagarajan, George Kantor, and Ralph Hollis. Integrated planning and control for graceful navigation of shape-accelerated underactuated balancing mobile robots. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 136–141, St. Paul, USA, 2012. xiii, xiv, xvii, xviii, 4, 22, 82, 83, 84, 92, 100, 101, 102, 104, 105, 113

[79] Umashankar Nagarajan, George Kantor, and Ralph Hollis. Integrated motion planning and control for graceful balancing personal robots. *International Journal of Robotics Research*, 2012. (Under Review). xiii, xiv, xvii, xviii, xix, 3, 4, 22, 80, 81, 83, 84, 88, 92, 94, 100, 101, 102, 103, 104, 105, 106, 107, 108, 113

[80] Umashankar Nagarajan, George Kantor, and Ralph Hollis. The ballbot: An omnidirectional balancing mobile robot. *International Journal of Robotics Research*, 2012. (Under Review). xiii, xiv, xv, 4, 18, 19, 20, 22, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40

[81] Umashankar Nagarajan, Byungjun Kim, and Ralph Hollis. Planning in high-dimensional shape space for a single-wheeled balancing mobile robot with arms. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 130–135, St. Paul, USA, 2012. xiv, xv, xvi, xix, 20, 21, 25, 60, 61, 62, 63, 64, 65, 66, 67, 68, 75, 78, 80, 82, 113, 119

[82] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1964. 55

[83] H. G. Nguyen, J. Morrell, K. Mullens, A. Burmeister, S. Miles, N. Farrington, K. Thomas, and D. Gage. Segway robotic mobility platform. In *SPIE Proc. 5609: Mobile Robots XVII*, Philadelphia, PA, October 2004. 3, 11, 48

[84] R. W. O'Flaherty, R. G. Sanfelice, and A. R. Teel. A hybrid control strategy for robust global swing-up of the pendubot. In *Proc. American Control Conference*, pages 1424–1429, 2008. 13

[85] Reza Olfati-Saber. Nonlinear control and reduction of underactuated systems with symmetry II: Unactuated shape variables case. In *Proc. 40th IEEE Conference on Decision and Control*, pages 4164–4169, 2001. 44, 46

[86] Reza Olfati-Saber. *Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles*. PhD thesis, Massachusetts Institute of Technology, February 2001. 12

[87] G. Oriolo and Y. Nakamura. Control of mechanical systems with second-order nonholonomic constraints: underactuated manipulators. *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, 3:2398–2403, 1991. 12

[88] Giuseppe Oriolo and Yoshihiko Nakamura. Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators. In *Proc. 30th IEEE Conference on Decision and Control*, volume 3, pages 2398–2403, 1991. 24, 26, 48

[89] James Patrick Ostrowski. *The mechanics and control of undulatory robotic locomotion*. PhD thesis, California Institute of Technology, 1996. 13

[90] Jim Ostrowski and Joel Burdick. Geometric perspectives on the mechanics and control of robotic locomotion. In *In Proc. International Symposium on Robotics Research*, pages 487–504. Springer Verlag, 1995. 13

[91] Jim Ostrowski and Joel Burdick. The geometric mechanics of undulatory robotic locomotion. *International Journal of Robotics Research*, 17:683–701, 1996. 13

[92] J.P. Ostrowski. Computing reduced equations for robotic systems with constraints and symmetries. *Robotics and Automation, IEEE Transactions on*, 15(1):111 –123, 1999. 13

[93] S. Patel, S-H. Jung, J. P. Ostrowski, R. Rao, and C. J. Taylor. Sensor based door navigation for a nonholonomic vehicle. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, pages 3081–3086, 2002. 14

[94] Mihail Pivtoraiko and Alonzo Kelly. Efficient constrained path planning via search in state lattices. In *8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2005. 80

[95] Mihail Pivtoraiko, Ross A. Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009. 80

[96] A. E. Quaid and A. A. Rizzi. Robust and efficient motion planning for a planar robot using hybrid control. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, pages 4021–4026, 2000. 14

[97] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, Rob Playter, and the BigDog Team. Bigdog, the rough-terrain quadraped robot. In *In Proc. 17th World Congress of the International Federation of Automatic Control*, pages 10822–10825, 2008. 3

[98] Muruhan Rathinam and Richard M. Murray. Configuration flatness of lagrangian systems underactuated by one control. 1998. 12

[99] J. R. Ray. Nonholonomic constraints. *American Journal of Physics*, 34:406–408, 1966. 5

[100] Rezero. http://www.rezero.ethz.ch/project_en.html. 11

[101] A. A. Rizzi. Hybrid control as a method for robot motion programming. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, volume 1, pages 832–837, 1998. 14

[102] A. A. Rizzi, J. Gowdy, and R. L. Hollis. Distributed coordination in modular precision assembly systems. *The International Journal of Robotics Research*, 20(10):819–838, 2001. 14

[103] Brandon Rohrer, Susan Fasoli, Hermano Igo Krebs, Richard Hughes, Bruce Volpe, Walter Frontera, Joel Stein, and Neville Hogan. Movement smoothness changes during stroke recovery. *Journal of Neuroscience*, 22(18):8297–8304, 2002. 6

[104] Per Rutquist and M. M. Edvall. *PROPT - Matlab Optimal Control Software*. Tomlab Optimization Inc., Pullman, WA, USA, 2010. 59, 112

[105] R. G. Sanfelice and A. R. Teel. A "throw-and-catch" hybrid control strategy for robust global stabilization of nonlinear systems. In *Proc. American Control Conference*, pages 3470–3475, 2007. 13

[106] Agostino De Santis, Bruno Siciliano, Alessandro De Luca, and Antonio Bicchi. An atlas of physical human-robot interaction. *Mechanism and Machine Theory*, 43(3):253–270, 2008. 38

[107] Eric M. Schearer. Modeling dynamics and exploring control of a single-wheeled dynamically stable mobile robot with arms. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, 2006. Report CMU-RI-TR-06-37. 17

[108] Elie Shammas, Howie Choset, and Alfred Rizzi. Towards automated gait generation for dynamic systems with non-holonomic constraints. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, pages 1630–1636, 2006. 13

[109] Elie A. Shammas, Howie Choset, and Alfred Rizzi. Natural gait generation techniques for principally kinematic systems. In *Proceedings of Robotics: Science and Systems*, June 2005. 13

[110] Elie A. Shammas, Karen Schmidt, and Howie Choset. Natural gait generation techniques for purely mechanical systems. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3664–3669, 2005. 13

[111] Elie A. Shammas, Howie Choset, and Alfred A. Rizzi. Towards a unified approach to motion planning for dynamic underactuated mechanical systems with non-holonomic constraints. *International Journal of Robotics Research*, 26:1075–1124, October 2007. 13

[112] Elie A. Shammas, Howie Choset, and Alfred A. Rizzi. Geometric motion planning analysis for two classes of underactuated mechanical systems. *International Journal of Robotics Research*, 26:1043–1073, October 2007. 13

[113] A. Shiriaev, J. W. Perram, and C. Canudas de Wit. Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach. *IEEE Transactions on Automatic Control*, 50(8):1164–1176, 2005. 12

[114] A. Shiriaev, L. B. Freidovich, and I. R. Manchester. Can we make a robot ballerina perform a pirouette? orbital stabilization of periodic motions of underactuated mechanical systems. *Annual Reviews in Control*, 32:200–211, 2008. 12

[115] A. Shiriaev, L. B. Freidovich, and S. V. Gusev. Transverse linearization for controlled

mechanical systems with several passive degrees of freedom. *IEEE Transactions on Automatic Control*, 55(4):893–906, 2010. 12

[116] Reid Simmons, J. Fernandez, R. Goodwin, S. Koenig, and Joseph O'Sullivan. Xavier: An autonomous mobile robot on the web. *Robotics and Automation Magazine*, 1999. xiii, 2

[117] Ole Jakob Sørdalen. Conversion of the kinematics of a car with n trailers into a chained form. In *IEEE International Conference on Robotics and Automation*, pages 382–387, 1993. 12

[118] Mark W. Spong. The control of underactuated mechanical systems. In *First International Conference on Mechatronics*, Mexico City, 1994. 5, 12, 43, 47, 137, 138, 141

[119] M.W. Spong. Partial feedback linearization of underactuated mechanical systems. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 1, pages 314–321, 1994. 12

[120] M.W. Spong. The swing up control problem for the acrobot. *Control Systems Magazine, IEEE*, 15(1):49–55, Feb 1995. 12

[121] Anthony (Tony) Stentz. The D* algorithm for real-time planning of optimal traverses. Technical Report CMU-RI-TR-94-37, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, October 1994. 115

[122] M. Stilman, J. Olson, and W. Gloss. Golem Krang: Dynamically stable humanoid robot for mobile manipulation. In *IEEE Int'l Conf. on Robotics and Automation*, pages 3304–3309, 2010. 11

[123] Y. Takahashi, S. Ogawa, and S. Machida. Step climbing using power assist wheel chair robot with inverse pendulum control. In *IEEE Intl. Conf. on Robotics and Automation*, pages 1360–65, 2000. 11

[124] R. Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In *Proc. Robotics: Science and Systems IV*, June 2009. 16

[125] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *International Journal of Robotics Research*, 29(8), 2010. 16, 116

[126] K. Teeyapan, J. Wang, T. Kunz, and M. Stilman. Robot limbo: Optimized planning and control for dynamically stable robots under vertical obstacles. In *IEEE Int'l Conf. on*

Robotics and Automation, pages 4519–4524, 2010. 11

[127] Sebastian Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, Frank Dellaert, Dieter Fox, D. Haehnel, Chuck Rosenberg, Nicholas Roy, Jamieson Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation (ICRA'99)*, 1999. xiii, 2

[128] M. M. Tobenkin, I. R. Manchester, and R. Tedrake. Invariant funnels around trajectories using sum-of-squares programming. In *Proc. 18th IFAC World Congress*, 2011. 16, 116

[129] S. Tsai, E. Ferreira, and C. Paredis. Control of the gyrover: A single-wheel gyroscopically stabilized robot. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, October 1999. 35

[130] O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, 1992. 58, 112

[131] Oskar von Stryk. Numerical solution of optimal control problems by direct collocation. In *in Optimal Control, (International Series in Numerical Mathematics 111*, pages 129–143, 1993. 58, 112

[132] H. Wang and et al. An experimental method for measuring the moment of inertia of an electric power wheelchair. *Proc. 29th Annual Int'l. Conf. of IEEE EMB*, pages 4798–4801, 2007. 27

[133] Jingang Yi, Yizhai Zhang, and Dezhen Song. Autonomous motorcycles for agile maneuvers, part i: Dynamic modeling. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 4613–4618, 2009. 12

[134] Jingang Yi, Yizhai Zhang, and Dezhen Song. Autonomous motorcycles for agile maneuvers, part ii: Control systems design. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 4619–4624, 2009. 12

[135] Mingjun Zhang and Tzyh-Jong Tarn. Hybrid control of the pendubot. *Mechatronics, IEEE/ASME Transactions on*, 7(1):79 –86, 2002. 13