An Agile and Dexterous Balancing Mobile Manipulator Robot

Roberto Shu

CMU-RI-TR-YY-NN

January, 2022

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Dr. Ralph Hollis, Carnegie Mellon University (Chair)
Dr. Nancy Pollard, Carnegie Mellon University
Dr. Oliver Kroemer, Carnegie Mellon University
Dr. Patrick Wensing, University of Notre Dame

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

Keywords: Ballbot, Balancing Robots, Mobile Manipulation, Underactuated Systems, Planning and Control

Abstract

This thesis focuses on designing and controlling a dynamically stable shape-accelerating dual-arm mobile manipulator, the Carnegie Mellon University (CMU) ballbot. The CMU ballbot is a human-sized dynamically stable mobile robot that balances on a single spherical wheel. We describe the development of a pair of seven-degree-of-freedom (DOF) humanoid arms. The new 7-DOF arm pair have human-like kinematics, a large bimanual workspace, and the strength to carry a 20 kg payload. As part of this thesis work, the pair of strong and lightweight arms are integrated into the CMU ballbot. To the best of our knowledge, this robot configuration is the first and only of its kind.

The ballbot class of robots is inherently unstable and requires careful coordination between the upper and lower body to maintain balance while performing manipulation tasks. This thesis also demonstrates that this new configuration for mobile manipulation can be controllable over a wide envelope of possible configurations. Two different control strategies are presented: (i) a decoupled lower and upper body control strategy where the existing balancing controller compensates for the arm movement while the arms react to the body motion; and (ii) an optimal whole-body planning and control strategy that considers the entire kinematics and dynamics of the system in a single formulation.

The CMU ballbot already has an existing robust balancing controller. This controller was designed for the ballbot in its initial configuration without arms. It also assumes the center of mass (COM) is aligned with the central axis of the ballbot's cylindrical body. To balance the robot, the controller tracks a zero body lean angle. With the addition of the new arms, the COM will constantly move off-axis, and the zero body lean angle will not stabilize the system. Here we explore the control of the entire system's COM by controlling the body lean angle. This strategy decouples the balancing task from the upper body motions. We present joint space and task space controllers to control the new arms. This decoupled control strategy is evaluated through experiments on the CMU ballbot.

Most solutions to dynamic whole-body motion planning and control either use a complex, full-body nonlinear dynamic model of the robot or a highly simplified robot model. Here we explore centroidal dynamics, which has recently become a popular approach for designing balancing controllers for humanoid robots. We describe a framework where we first solve a trajectory optimization problem offline and later use the same nonlinear program (NLP) with a shorter time horizon in a model predictive control (MPC) context to execute the motion. We define balancing for a ballbot in terms of the centroidal momentum instead of other approaches like zero moment point (ZMP) or angular velocity that are more commonly used. We demonstrate that this framework can generate combined loco-manipulation motion plans and control inputs for the CMU ballbot.

Acknowledgments

The journey of a PhD can sometimes seem to be a lonely path. However, it would be impossible to achieve all the work and effort required to complete one without all people that surround you. I would like to thank all those who have helped me in this endeavor.

I would like to express my deepest gratitude to my advisor, Ralph Hollis, with whom this work would have not been possible. You took a chance on me and gave me the opportunity to work on a very unique robot, the ballbot. It has been a true honor and pleasure to work with you, and I have learned so much from you on how to build robots. You keep inspiring me with your ambition, drive, and commitment to the field of robotics and life.

I want to thank my committee member, Nancy Pollard, for all her valuable comments and discussions throughout our time working together. I would also like to thank Oliver Kroemer for his willingness to work hands-on with the ballbot and for helping me organize my thoughts. I would also like to thank my external committee member, Patrick Wensing, for his valuable comments and insights on robot control theory and implementation.

I have had the pleasure of working alongside some brilliant minds. I want to thank Fabian Sonnleitner for all the work we did together on the ballbot; you helped me push the limits of the ballbot. To Shreyas Srivatchan, Cornelia Bauer, Dominik Bauer, and Saumya Saxena, thank you for helping me run experiments and collaborate with me on so many different ideas. I also want to thank all the many students that worked on the ballbot before me. Without your contributions to the ballbot project, this work would not have been possible.

I also want to thank all my friends at the RI and in Pittsburgh. Thank you for sharing this journey with me. Without you, it would not have been as fun.

Last but not least, I want to express my immense gratitude to my family. To my best friend Juan Felipe, you have been in the highs and lows cannot thank you enough. To my girlfriend, Paola, thank you for your unconditional support and for encouraging me through times of adversity. Without the countless days and nights, you spent in the lab helping me run experiments, I would not have been able to reach the finish line.

To my grandma, you have always believed in my abilities and taught me that hard work pays off through example. My siblings Johanna, Adrian, and Leonardo, your encouragement and advice have been indispensable throughout this journey. You are always the first to like and share any posts of my work. To my brother-in-law Pedro, without you, there would be no ballbot logo; thank you. To my nephew Benjamin, I hope that you realize your fullest potential when you grow up and work on something you are passionate about like I had the opportunity to do so.

To my parents, you have always been my biggest supporters, and I can not thank you enough for the opportunities you have given me to follow my hopes and aspirations. Without you, both this PhD would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The role of mobile robots in human environments is increasing every day. We see robots working with or alongside people in critical applications ranging from manufacturing to health care and quality of life. The increasing role of robots in human environments correspondingly increases the demand for their capabilities and functionalities. Mobile robots should have arms to manipulate objects and the environment to be truly useful. Mobile robot bases have been developed over many decades, but only recently have researchers added arms to these bases, opening up the rich field of mobile manipulation. To date, the robot configurations need wide, heavy bases to support the arms and provide stability. Such robot bases, in turn, severely limit maneuvering agility in cluttered human environments. Insufficient agility, dexterity, and intelligence have hindered mobile manipulators from being at a stage of real value to people. Despite exciting research progress, there remains a need for robots that can fully utilize the dynamics of their bodies to accomplish challenging tasks in an efficient, robust, and safe manner. This thesis work develops a unique vision for mobile manipulator robots that embodies a combination of three main attributes: (1) dynamic stability + (2) agile mobility + (3) dexterous interaction. The goal of this work is to develop foundational capabilities that, in the future, will enable the creation of a robot with an unprecedented ability to perform complex dynamic tasks with a complete and unified understanding of dynamics, control, software, hardware, and the environment.

1.1 Motivation

There exist two types of mobile manipulators, as shown in Fig 1.1. The most common configuration for human-sized mobile manipulators is to be statically stable. This robot class has wide, heavy bases that may or may not be omnidirectional. This is so to keep the center of mass (COM) as low as possible to provide stability. A high center of mass and/or small base will cause the robot to tip over easily, which is not desirable. Thus, these robot designs remain bulky, cumbersome, and lack the capability for truly agile motion. It is becoming increasingly apparent that to be most effective, mobile manipulators should be dynamically stable machines that actively balance just like humans do.

Unlike their statically stable counterparts, dynamically stable robots can be more effective personal robots as they can be tall enough for eye-level interaction, have a small footprint to

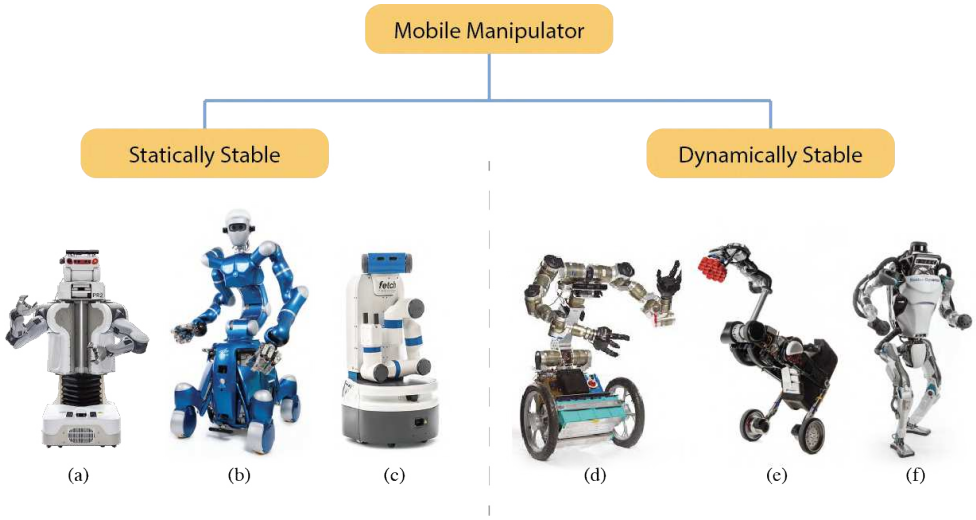


Figure 1.1: Examples of notable existing mobile robots: (a) Willow Garage PR2 [1], (b) DLR Rollin' Justin [2], (c) Fetch Robotics Mobile Manipulator [3] are statically stable. While (d) Golem Krang [4], Boston Dynamics (e) Handle [5] and (f) Atlas [6] are dynamically stable.

navigate cluttered environments, and accelerate and decelerate quickly. The ability to actively maintain postural stability makes this type of robot platform effective mobile manipulators. They can generate forces on external objects, withstand greater impact forces, and leverage their body dynamics. These traits make them ideal platforms for safe deployment in crowded spaces such as warehouses, homes, businesses, and retail environments, which the robots may share with people or in which they may cooperate with people.

Many common interactions utilize significant dynamics. For example, when opening a heavy door or initiating the motion of a wheelchair, people use the dynamic motion of their bodies to make the task easier. Few existing robots are capable of such abilities, especially while working with people. Exploiting dynamics to perform supportive or cooperative tasks is in contrast to many of the usual paradigms of manipulation and navigation today. Yet, this idea is critical for the safe deployment of robots in crowded spaces. The characteristics of dynamically stable mobile robots give them the potential to realize such dynamic and agile tasks. Real-time control of the desired actions requires innovation in simultaneous execution of different subtasks such as maintaining balance, being compliant to external forces, and ensuring accurate end-effector position and force tracking.

In this thesis work, we explore the development of a new type of dynamically stable agile and dexterous mobile manipulator by adding a pair of 7-DOF arms and multi-DOF hands to a ballbot, as illustrated in Fig 1.2. The CMU ballbot research platform [12, 13] was the first robot that dynamically balanced and locomoted on a ball, shown in Fig 1.3. It is an underactuated system that accelerates by leaning its body, but cannot directly control its lean angle. This makes motion planning and controls nontrivial but yields many benefits. The ballbot class of mobile robots provides unique advantages over a legged or multi-wheeled counterpart; they can be tall enough to interact with people at eye level and slender enough to easily move around in cluttered human environments. The single spherical wheel provides a small footprint and omnidirectional

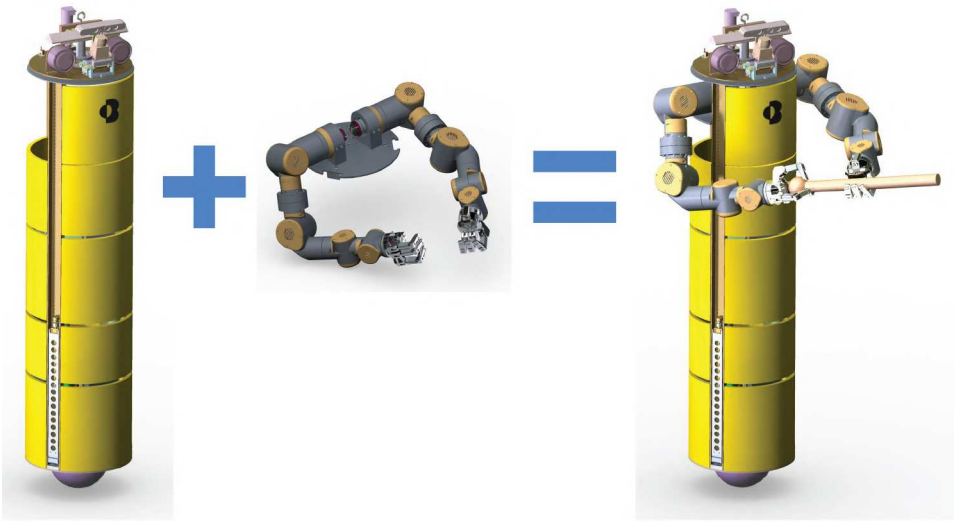


Figure 1.2: Proposed mobile manipulator: (a) CAD model of existing CMU ballbot with simple 2-DOF arms removed; (b) pair of 7-DOF arms with hands to be developed; (c) resulting ballbot research platform with dexterous arms and hands shown manipulating a simple object

motion, making it difficult for the ballbot to get trapped in tight places. Most importantly, owing to the action of their balancing controllers, ballbots have *intrinsic omnidirectional compliance* enabling soft, gentle interaction with people. The CMU ballbot can be easily moved by the force of a single finger yet cannot be upset by a strong push. This combination of desirable features does not exist in traditional mobile robots. A detailed description of the ballbot is available in Chapter 2.

Ballbots present many interesting dynamics and control challenges but yield many benefits, especially for physical human-robot interaction (pHRI) [14]. An interesting research avenue is to add arms to the ballbot. A pair of simple two degrees of freedom (DOF) arms have already been developed for the ballbot [15]. Reliable balancing and navigation over long distances with the arms have been demonstrated [15]. The limitations of this arrangement are obvious. The simple 2-DOF arms [16, 17] shown in Fig. 1.3(b)-(d) do not have a human-like appearance, size, and manipulative capabilities which hinders operation in human environments where furniture, tools, and objects are optimized for the human anatomy [9]. However, this provides an excellent starting point for the work in this thesis because as a proof of concept it has established feasibility. By equipping the ballbot with far more capable 7-DOF arms and multi-DOF hands, it will be possible to grasp and manipulate objects using one or both arms, carry the objects over arbitrary distances at high speed, and place or assemble them into larger structures. It will also be possible to interact dynamically with larger aspects of the environment for tasks such as opening and closing doors or pushing/pulling on objects such as furniture while using both its body lean and arm strengths. There are other ballbots being developed around the world [18, 19, 20, 21, 22], as shown in Fig. 1.4. However, to our knowledge, besides the CMU ballbot only one other ballbot has been shown to operate with an arm mounted on it. That robot is Rezero [23], it is roughly

Figure 1.3: Views of the CMU ballbot with simple arms shown dynamically balancing: (a) photo of the ballbot with body panels navigating autonomously; (b) arms in neutral pose; (c) arms outstretched with up to 2 kg weights; (d) arms waving around while balancing and maintaining position on the floor (Courtesy of Ralph Hollis).

half the size of the CMU ballbot and has a single 3-DOF arm mounted on top of it. The work in this thesis will result in the first-of-its-kind ballbot robot with a pair of 7-DOF arms.

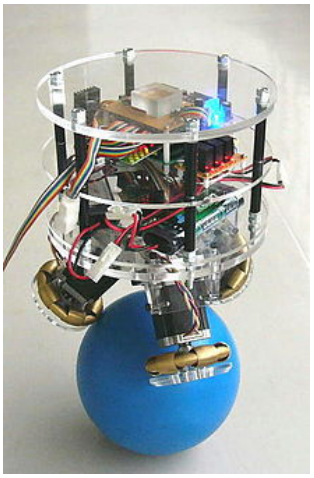
1.2 Thesis Objectives

It has been successfully shown that balancing mobile robots, like the ballbot, have the dynamic capabilities to navigate human environments with speed and grace comparable to that of humans. However, comparable manipulation capabilities are still to be demonstrated. This thesis is focused on proving the hypothesis that the resulting first-of-its-kind, highly dynamic ballbot with 7-DOF arms is controllable over a wide envelope of possible configurations. Within this controllability envelope, it is necessary to prove that purposeful actions can be successfully planned and carried out. Basic capabilities to be demonstrated include, *e.g.*, accurately moving the arms/hands to points in space while dynamically balancing; grasping objects fixed in the environment; lifting objects of unknown mass while adaptively compensating for their effects; robust mobility and navigation while moving the arms and holding objects. To achieve these results three main objectives have been set for this thesis work.

TO 1. *Build a first-of-its-kind platform for mobile robot manipulation by enhancing the CMU ballbot mobile platform by adding a pair of custom-built 7-DOF arms and multi-DOF hands.*

TO 2. *Develop a robust control framework for the enhanced ballbot that will enable it to successfully plan and execute motions over a wide envelope of possible configurations.*

TO 3. *Demonstrate that within the controllable envelope, the robot can carry out purposeful actions in a safe and effective manner.*



BallIP [18]

Rezero [20, 23]

Kugle [22]

Figure 1.4: Other existing ballbots around the world used for academic research.

Achieving these three objectives will constitute a unique new platform for *agile and dexterous mobile manipulation*. Here, the term *agile* refers to the ability to smoothly move through the environment while avoiding static and dynamic obstacles. The term *dexterous* refers to the ability to skillfully manipulate objects and interact with the environment.

1.3 Approach

To achieve the goals of this thesis requires innovative solutions to hardware, planning algorithms, and real-time control for the safe, effective operation of the dynamic multi-DOF mobile manipulator proposed.

Hardware Development

Reaching the compound goal of developing an agile and dexterous mobile manipulator will be enabled by creating a pair of new lightweight but strong human-scale arms/hands for the ballbot. Ideally, one would purchase such arms and interface them to the ballbot. Unfortunately, no existing off-the-shelf 7-DOF arms are available that meet the requirements of scale, weight, and actuation strength needed for the enhanced ballbot. The goal is to enable the enhanced ballbot to lift and carry up to 20 kg loads with both arms while reacting compliantly. Accordingly, this thesis presents the design of the arms based on modular integrated actuators that combine a motor, drive electronics, harmonic gearing, torque, position sensing, and a network interface in a compact lightweight package. Structural joints and lightweight links were designed and fabricated. To complement the existing compliance of the ballbot, additional compliance for the arms is generated by having a lightweight structure combined with high-resolution torque sensing. Integrating the new 7-DOF arms to the CMU ballbot opens a wide range of new research areas.

Balancing by Regulating Body Lean Angle

The CMU ballbot already has a robust and effective balancing controller [24]. It is robust against external disturbances and model errors. Since the ballbot is underactuated, the controller employs an outer and inner loop structure. The inner loop tracks a desired body lean angle. The outer loop generates a body lean angle trajectory to follow a desired ball position on the floor. The controller was designed for the ballbot without arms. Without modification, the controller cannot stabilize the ballbot with the new 7-DOF arms. To mitigate this problem, the work in this thesis presents an extension to the balancing controller. The extension enables the control of the entire system's COM via regulating the robot's body lean angle. This approach shifts the old control inputs to operate about the actual equilibrium point instead of the zero equilibrium point.

Whole-Body Motion Planning

A typical approach to controlling mobile manipulators is to use an inverse dynamics control strategy. This method achieves satisfying results for fully actuated statically stable mobile manipulators. However, for *underactuated shape-accelerating* systems, like the ballbot, inverse dynamics control is more difficult to apply because of the presence of unstable zero dynamics [23]. This thesis work presents an optimization-based whole-body planning framework. The whole-body planner enables the careful coordination required between the upper and lower body of the robot to maintain balance while performing manipulation tasks. Trajectory optimization forms the basis of the motion planner. In the formulation of the trajectory optimization problem, the robot's dynamics are represented via the system's centroidal dynamics with kinematic constraints. This avoids the use of the complex non-linear dynamics of the system. This model representation is complex enough to exploit the full-body dynamics of the system but is also simple enough to have a low computational burden. Moreover, inspired by momentum-based-balancing controllers [25] postural balance is defined in terms of the centroidal momentum instead of traditional approaches like ZMP or angular velocity.

Optimal Model-Based Control

Using a Model Predictive Control-like approach can make the system's control substantially more robust, facilitating a fast and safe reaction to unknown external disturbances and model uncertainty. This approach can improve safe operation around humans. This thesis investigates modifying the optimization problem used for planning in an MPC context. In the MPC formulation, a hierarchical structure is defined. Higher priorities are set to primary tasks (*e.g.*, balancing and self-collision avoidance), and lower priority to secondary tasks (*e.g.*, reaching for an object with one end-effector while reaching for a second object with another end-effector) while guaranteeing the execution of primary tasks. Experiments in simulation and hardware are conducted to evaluate the proposed planning and control framework.

1.4 Thesis Outline

The rest of this document is organized as follows:

Chapter 2 introduces the Carnegie Mellon University (CMU) ballbot research platform. The history of the platform and a description of the hardware and software architecture of the system are presented.

Chapter 3 investigates a kinematic-based center of mass compensation control scheme to maintain balance while performing manipulation tasks that require heavy payload lifting on the ballbot with its simple 2-DOF arms. A method to locomote while carrying heavy loads is also presented. The controller effectiveness is evaluated by demonstrating different payload transportation and robot-to-human handovers. The chapter results serve as proof that the ballbot with 7-DOF arms is feasible.

Chapter 4 begins with the design and development of a new 14-DOF dual manipulation system to replace the existing 2-DOF arms in the CMU ballbot research platform. Joint and task space control algorithms implemented on the arms are also presented. Successful integration of the arms to the CMU ballbot is demonstrated by executing heavy payload manipulation experiments and compliant interaction with humans. The existing balancing controller is extended to enable the ballbot to balance while operating the 7-DOF arms. Successful experiments of the ballbot balancing while moving the arms are shown.

Chapter 5 presents a task space impedance control formulation to control the ballbot arms. It extends the traditional implementation of task space impedance control for fixed-based fully actuated robot manipulators to dynamically stable underactuated mobile manipulators. Experimental results to evaluate the controller performances are shown alongside different demonstrations of the controller in action.

Chapter 6 presents a planning and control framework for dynamic, whole-body motions for dynamically stable shape-accelerating mobile manipulators. Centroidal dynamics, which has recently become a popular approach for designing balancing controllers for humanoid robots, is explored. The framework is demonstrated to be capable of generating dynamic motion plans. A model-based control formulation is presented to track the generated motions plan. The framework is evaluated in a simulation and the real CMU ballbot with 7-DOF arms.

Chapter 7 concludes the thesis with a summary of the contributions made by the work in this thesis and presents potential future work.

Chapter 2

The ballbot

This thesis focuses on the Carnegie Mellon University (CMU) ballbot research platform. This mobile robot is a dynamically-stable robot that balances and locomotes on a single spherical wheel. Ballbot type robots offer unique advantages over other mobile robots and present interesting dynamics and control challenges. This chapter introduces the CMU ballbot. The physical system, hardware, sensor components, and software architecture are presented.

2.1 Background

The CMU ballbot [12, 13] was the first successful dynamically stable mobile robot that balances on a single spherical wheel. The ballbot was invented by Dr. Ralph Hollis in 2005 at the Microdynamics Systems Laboratory (MSL) at Carnegie Mellon University [26]. Unlike Segway-type two-wheel balancing robots, the ballbot is omnidirectional. It can move in any direction without first turning its body. It is an underactuated system that accelerates by leaning but cannot directly control its lean angle. This property makes motion planning and control nontrivial, but yields many benefits, especially for physical Human-Robot Interaction (pHRI) [27]. The ballbot class of mobile robots provides unique advantages over a statically-stable robot such as a tall body enabling human-robot interaction at eye level; a small footprint afforded by the single ball wheel; and omnidirectional motion with *intrinsic omnidirectional compliance* afforded by its balancing controller. The CMU ballbot was designed to be of human proportions to operate in human environments.

Early work by Tom Lauwers [12, 28] enabled the ballbot to balance and station keep. Further capabilities were developed to automatically transition between its statically stable state and its dynamically stable state by Anish Mampetta and Umashankar Nagarajan [29, 30].

This thesis builds on the fundamental work by Eric Shearer in 2006 that modeled dynamics and control of a ballbot that would have a pair of simple 2-DOF arms [31]. In 2012, a pair of 2-DOF series-elastic arms were developed and interfaced to the ballbot. Byungjun Kim and Umashankar Nagarajan [15] demonstrated stable motion planning and execution while moving the arms. The task was to control the ballbot's ball position on the floor utilizing the arm motion. Precise end-effector control was not shown.

Nagarajan performed early exploration of the ballbot as a platform for pHRI, where the ball-

bot inferred the difference between a soft push and a shove. This allowed physical guidance along a path and then autonomously repeat the path [14]. In 2010, he introduced a hybrid control formalism for navigation of shape-accelerated balancing systems [16, 32]. He extended his work in 2013 and developed a comprehensive hybrid control strategy for ballbot navigation [17]. A complete description of the CMU ballbot's capabilities using this strategy was presented in 2014 [13].

In 2012, Michael Shomin formulated a differentially flat representation of the ballbot system; this enabled fast analytic trajectory planning [33]. He extended his work to develop a comprehensive differential flatness-based planning strategy for the CMU ballbot to enable autonomous navigation through cluttered indoor environments with static and dynamic obstacles [34], as shown in Figure 2.1(a). In 2015, Shomin extended the pHRI capabilities of the CMU ballbot and its 2-DOF arms by demonstrating the robot can assist in helping a person in sit-to-stand maneuvers by exerting forces up to 120 N by leaning up to 15° from vertical [7], as shown in Fig. 2.1(b). Leveraging the CMU ballbot's navigational capabilities, he presented a method to lead people by the hand physically. A human subject trial was conducted demonstrating the ballbot successfully leading participants to multiple goals utilizing an amount of force that users found comfortable [8], as shown in Fig. 2.1(c). The same year, Bhaskar Vaidya developed a compensation strategy for ballbots operating with center-of-mass offsets and on Americans with Disabilities Act (ADA)-compliant ramps [35]. The approach builds on top of the existing balancing controller by introducing feedforward compensating body lean angle commands. This thesis follows a similar strategy to balance with the new 7-DOF arms.

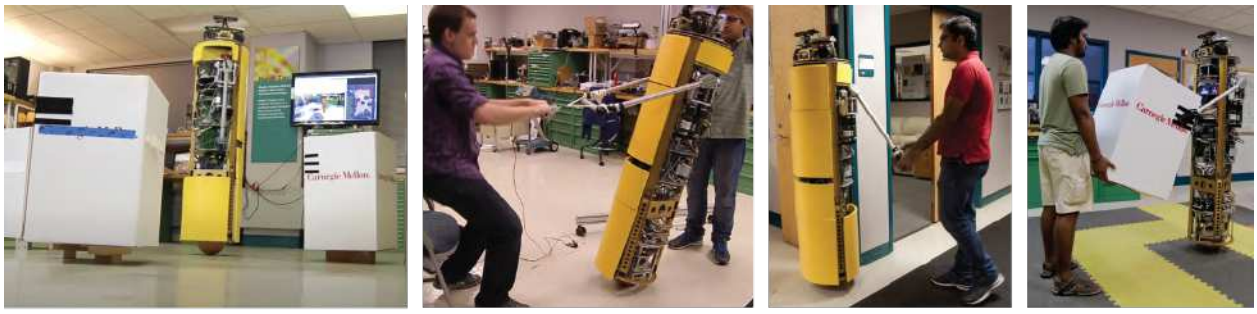


Figure 2.1: Photos of the ballbot using a pair of simple 2-DOF arms: (a) autonomously navigating; (b) providing up to 120 N of assistive force in a sit-to-stand maneuver [7]; (c) leading a person through a doorway; (c) cooperative carrying of a bulky object.

2.2 System Description

Before this thesis, the CMU ballbot consisted of a 1.71 m tall and 0.368 m wide cylindrical body on top of a 185 mm radius ball. The robot had a total weight of 65 kg. The ballbot had a pair of strong, compliant 2-DOF arms. Each of these arms was made of a single 0.56 m straight

hollow aluminum tube with a rubber knob as hand. Two series-elastic actuators actuated each arm that controls the shoulder adduction/abduction and flexion/extension joints. The robot was intentionally built to be of a human-size to interact with human environments, similar to the way humans do. Fig. 2.2 depicts the evolution of the CMU ballbot research platform from its inception in 2006 (Fig. 2.2(a)), to its state in 2010 when the simple 2-DOF arms were integrated ((Fig. 2.2(b)), to its current form with the enhancement made in this thesis work (Fig. 2.2(c)).

(a) (b) (c)

Figure 2.2: The dynamically stable mobile robot CMU ballbot in its different development stages through the years, circa (a) 2006, (b) 2010, and (c) 2019 (this thesis work).

The CMU ballbot is a self-contained semi-autonomous mobile robot with all the required components for operation onboard. Batteries, charger, computers, inertial measuring unit (IMU), and a "light detection and ranging device" (lidar) are located in the body. The lidar is used for localization and navigation within a map.

The ballbot drives the ball wheel on which it balances through its four-motor *Inverse Mouse Ball Drive* (IMBD) [12]. This mechanism works in the inverse fashion of an old trackball computer mouse. The ball is squeezed by four orthogonal rollers that are actuated by DC servomotors via pulleys and belts, as shown in Fig. 2.3. Each pair of actuated opposing rollers drive the ball in each of the two orthogonal motion directions on the floor [30]. This mechanism achieves omnidirectional motion by moving the ball in any direction. The ballbot can travel at speeds up to 1.5 m/s over long distances indoors while dynamically balancing. Turning the body is sometimes necessary to orient sensors on the body to a task or position its arms. Thus, the IMBD 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). Another DC servomotor actuates this yaw degree of freedom. A slip ring assembly enables unlimited yaw rotation of the body.

At the top of the body is a pan-and-tilt sensory turret with multiple sensors to perceive and interact with the environment, as shown in Fig. 2.4. The turret includes an ASUS Xtion RGB-D

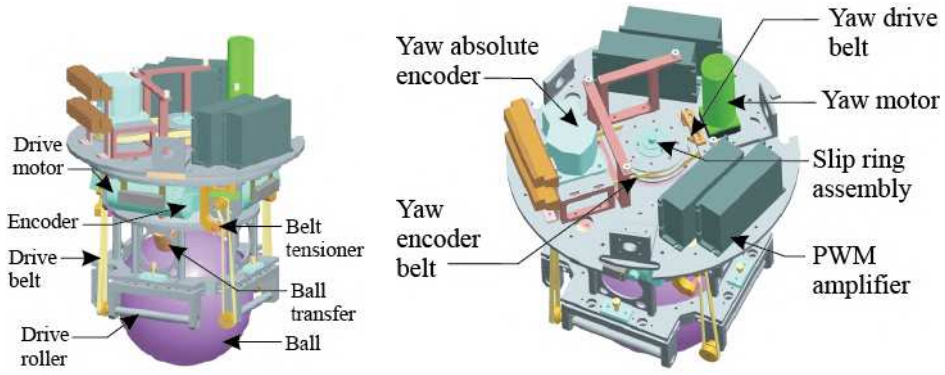


Figure 2.3: CAD model of the CMU ballbot's Inverse Mouse Ball Drive (IMBD) mechanism (Courtesy of Ralph Hollis).

sensor, a Hokuyo UST-05LN 5 m laser rangefinder, an Acoustic Magic directional microphone, and two speakers. The turret can pan 700° and can tilt the RGB-D sensor and laser range finder 90° to face straight down. The ballbot is capable of helpful verbal interaction with humans. The microphone and speaker enable the robot to receive spoken commands and provide speech feedback of its state and actions.

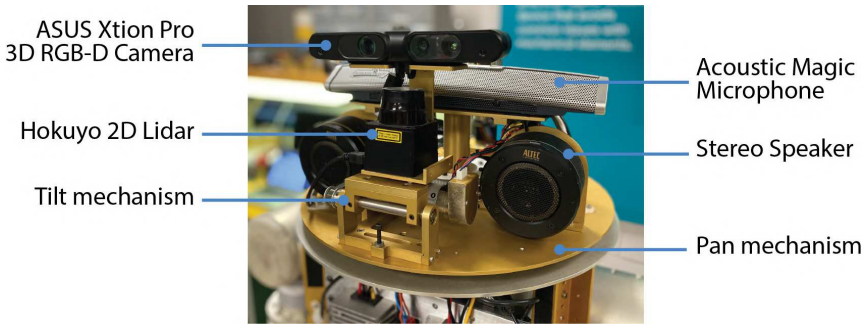


Figure 2.4: The ballbot's pan-and-tilt sensory turret with 3D camera, laser rangefinder, microphone and speakers.

The robot also houses a triad of legs that descend from the body to automatically transition from dynamically stable (DS) (*i.e.*, balancing) to a statically stable state (SS) and vice versa [30]. When the robot's legs are deployed, it can stop operation to charge its 4 12V sealed lead-acid batteries. These batteries allow the robot to run for approximately 1.5 hours. A video summarizing the existing capabilities of the ballbot can be found on YouTube[1].

[1] https://www.youtube.com/watch?v=8BtDuzu2WeI&t=2s

2.3 Computing and Software Architecture

2.3.1 Controller Design

The balancing controller, for obvious reasons, is the single most important controller on the robot. As discussed previously, the ballbot is an underactuated system. The ballbot's ball is actuated, but the body lean angles are unactuated. To control the ballbot's body, a cascading control scheme is implemented with an inner and outer loop as discussed extensively in [24]. A schematic of the balancing controller is shown in Fig. 2.5. The inner PID controller closes the loop around the

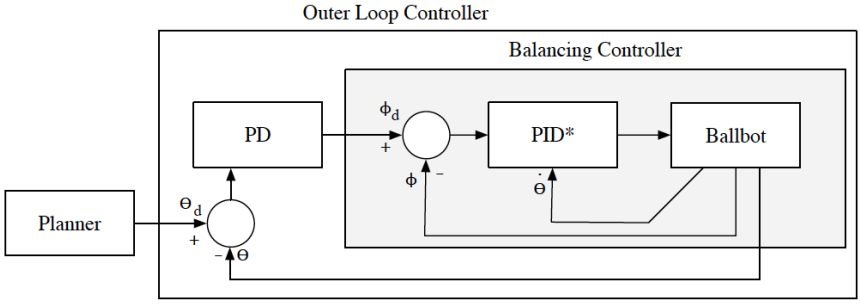


Figure 2.5: Diagram of the ballbot balancing controller [8]

desired lean angle, *i.e.* the roll and pitch angles of the body, and balances the robot about these desired body angles. The outer loop PD controller tracks the desired ball position trajectory on the floor. The outer loop provides body lean angle commands to the balancing controller. By tracking a desired lean angle, the ballbot can indirectly track a ball position. Although this may not yield the same performance as an inverse dynamics or LQR controller, having an inner loop balancer is a more robust approach as substantiated by experiments in [15]. This method has the advantage of placing limits on lean angles and ensuring that the robot will not fall over as long as the balancing controller is stable within the lean angle limits. The body controller is implemented as two independent controllers, one for each of the vertical planes.

The body lean angles are directly measured with the VectorNav VN-100 Inertial Measurement Unit (IMU) and Attitude Heading Reference System (AHRS). The body lean angles are measured with respect to the gravity vector with an accuracy of $\pm 0.5°$. The ball angles correspond to the rotation measured by the encoders on the IMBD motors.

The ballbot's yaw control is decoupled from the balancing controller. The yaw motor is controlled with PID feedback control on the yaw estimate taken from the IMU. The turret pan and tilt are controlled via PID controllers with feedback from encoders.

2.3.2 Software Architecture

All the software running on the CMU ballbot is distributed between two machines: a real-time system and a high-level non-real-time system. A third offboard "ground base station" computer

is used to visualize data, monitor data, send high-level commands, and log data. It connects wirelessly via Wi-Fi to the robot's onboard network.

The real-time machine is an Intel Core 2 Duo @ 2.4 Ghz running the QNX operating system. This machine is responsible for balancing the robot and control of the legs. The balancing control loop operates at 500 Hz, triggered by an RTC timer module. Non-real-time critical software packages are run on the high-level machine. As part of the work in this thesis, this computer has been upgraded to a Zontac Magnus EN72070V with an Intel Core i7 six-core 2.6 Ghz CPU and NVIDIA GeForce RTX 2070 GPU running Ubuntu 16.04. The addition of a GPU opens the possibility for computer vision and machine learning algorithms to run onboard. All software in the high-level machine is developed to be compatible with the Robotics Operating System (ROS) Kinetic middleware [36]. This allows code to be more modular, facilitates communication between different pieces of code, and enables the use of a vast amount of open-source robotics software packages.

The two onboard computers are connected via Ethernet and communicate using rosbridge [37], a serialization/deserialization package for ROS communication via a socket connection. The QNX system relays odometry, IMU data, and diagnostic information to the Ubuntu system. The Ubuntu machine sends desired lean angle, velocity, and state-of-operation commands to the controllers running in the QNX machine.

The high-level machine runs the localization, navigation, arms control, and communication with the base station components. The pair of 7-DOF arms are connected via Ethernet to this machine, which allows direct control at 250 Hz. The ROS Control framework [38] is used to control the arms, enabling the use of the many open-source grasping and manipulation packages. The high-level machine also controls the sensory turret and connects to all its sensors.

The ground base station computer controls the robot via high-level commands and logs data via the ROS network. A graphical user interface is available to toggle the primary function of the ballbot. This includes starting controllers, lifting the legs, start to balance, etc. RViz and RQT ROS tools are used to visualize, receive, and send data between the base station and the ballbot's high-level machine.

Chapter 3

Lifting Heavy Payloads

Ballbot type robots belong to the family of *shape-accelerated underactuated balancing systems* [32]. This means that non-zero changes in joint configuration result in accelerations in the position space of the base. Because of this, ballbots equipped with manipulators can use the additional degrees of freedom to keep balance, and better trajectory tracking [32, 39]. However, this is a curse when trying to lift heavy payloads. Large payloads at the distal end of the arms will cause the system's center of mass (COM) position to move away from the single support point (POS). In turn, the body will accelerate, and if not controlled, the system will become unstable. The existing balancing controller is not able to compensate for the large disturbances. One approach to mitigate this is to introduce additional commands to force the COM position to be on top of the single point of support.

This chapter presents a pragmatic kinematic-based COM compensation controller that allows balancing mobile robots, like the ballbot, to lift and transport heavy payloads safely and smoothly. This strategy was implemented and experimentally validated on the ballbot with 2-DOF arms. With the 2-DOF arms, the ballbot's payload carrying capacity increased from 2 kg [40] to 15 kg while dynamically balancing. The successful demonstration of carrying a heavy load with the 2-DOF proves that the ballbot can operate with a more complex and heavier pair of 7-DOF arms.

The work presented in this chapter with the 2-DOF arms was done in collaboration with Fabian Sonnleitner. Sections of this chapter appeared in [41].

3.1 Background

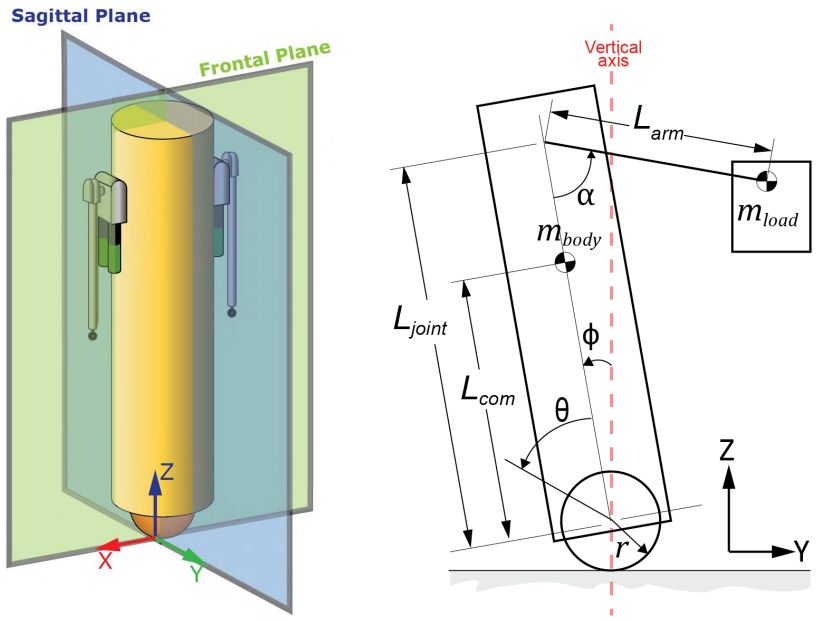
3.1.1 Underactuated Dynamical Systems

The force Euler-Lagrange equations of motion for a dynamical system in matrix form are given by:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{S}\boldsymbol{\tau}, \tag{3.1}$$

where n is the number of DOF of the system, $\boldsymbol{q} \in \mathbb{R}^n$ is the configuration vector, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the mass/inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal matrix, $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^n$ is the vector of gravitational forces, $\boldsymbol{\tau} \in \mathbb{R}^{n_c}$ is the vector of generalized input forces, and

$\boldsymbol{S} \in \mathbb{R}^{n \times n_c}$ is the actuation selection matrix that separates the n_c controlled and n_u uncontrolled DOF. A system satisfying Eq. 3.1 is said to be an *underactuated system* [42] if $n_c < n$, *i.e.*, there are fewer independent control variables than configuration variables. The ballbot with 2-DOF



(a) (b)

Figure 3.1: (a) Decomposition of 3D ballbot model into two orthogonal planes used for planning and control. (b) Planar ballbot model and notation diagram projected onto the Sagittal (Z-Y) plane.

arms has eight configuration variables given by $\boldsymbol{q} = [\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\alpha}]^T \in \mathbb{R}^8$, where $\boldsymbol{\theta} = [\theta_x, \theta_y]^T \in \mathbb{R}^2$ are the ball angles, $\boldsymbol{\phi} = [\phi_x, \phi_y]^T \in \mathbb{R}^2$ are the body lean angles, and $\boldsymbol{\alpha} = [\alpha_{l_x}, \alpha_{l_y}, \alpha_{r_x}, \alpha_{r_y}]^T \in \mathbb{R}^4$ are the left and right arm joint angles. As discussed in section 2.3.1 two instances of the balancing controller are implemented, one on each of the vertical planes shown in Fig. 3.1 (a). Consequently, we derive the dynamic using a planar representation of the ballbot with 2-DOF arms while carrying a payload as shown in Fig. 3.1 (b). For the systems the body configuration variables $\boldsymbol{\phi}$ are unactuated, while the ball $\boldsymbol{\theta}$ and arm $\boldsymbol{\alpha}$ configurations are actuated. Therefore, the system is underactuated, it cannot directly control the body lean angle. The ballbot with 2-DOF arms has six actuated degrees of freedom and two unactuated degrees of freedom, *i.e.* $n_c = 6$ and $n_u = 2$.

3.1.2 Shape-Accelerating Balancing Systems

The special class of underactuated systems called *shape-accelerating balancing systems*, for which the ballbot is part of, was introduced by Nagarajan [40]. This section provides only a summary of shape-accelerating balancing systems, for an in-depth description refer to [40]. The configuration variables $\boldsymbol{q} \in \mathbb{R}^n$ of a dynamic system can be split into position variables $\boldsymbol{q}_p \in \mathbb{R}^{n_p}$, and shape variables $\boldsymbol{q}_s \in \mathbb{R}^{n_s}$, *i.e.*, $\boldsymbol{q} = [\boldsymbol{q}_p, \boldsymbol{q}_s]^T$ and $n_p + n_s = n$. For the ballbot with arms,

the ball angles form the position variables, *i.e.*, $\boldsymbol{q}_p = \boldsymbol{\theta} \in \mathbb{R}^2$, whereas the body and arm joint angles form the shape variables, *i.e.*, $\boldsymbol{q}_s = [\boldsymbol{\phi}, \boldsymbol{\alpha}_l, \boldsymbol{\alpha}_r]$. The ballbot with 2-DOF arms has $n_p = 2$ position variables and $n_s = 6$ shape variables. Note that the ball angles relate to the position of the ball and not the actual orientation of the ball. The position $\boldsymbol{P}_b \in \mathbb{R}^2$ of the ball relative to the inertial frame is given by:

$$\boldsymbol{P}_b = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} r(\theta_x + \phi_x) \\ r(\theta_y + \phi) \end{bmatrix}, \tag{3.2}$$

where r is the radius of the ball.

Shape-accelerated balancing systems have the property that their shape configurations can be mapped to the accelerations of the position variables. These systems are destabilized by gravitational forces and have non-integrable constraints on their dynamics. Balancing mobile robots, such as the ballbot, are examples of such systems. For ballbot type robots this characteristic is in part due to having a very small patch of support (*i.e* approximately the size of a 25 cent coin, that we model as a single point of support) rather than a large polygon of support like multi-wheeled mobile bases. Consider the planar ballbot schematic in Fig. 3.2 and assume the robot starts in an equilibrium configuration (*i.e.*, $\boldsymbol{q}_s^{eq}$ for which $\ddot{\boldsymbol{q}} = 0$) as in Fig. 3.2 (a). Any non-zero change in shape configuration (e.g., moving an arm as in Fig. 3.2 (b)) will result in the COM of the system moving away from the point of support. Consequently, the system will accelerate in the direction of the center of mass displacement and make the system unstable, as shown in Fig. 3.2 (c). For the system to stabilize, it has to find a new equilibrium configuration $\boldsymbol{q}_s^{eq'}$ where the center of mass is on top of the point of support, such as leaning backward, as shown in Fig. 3.2 (d). This behavior worsens when carrying heavy payloads as the COM shift will be larger, causing the system to accelerate more.

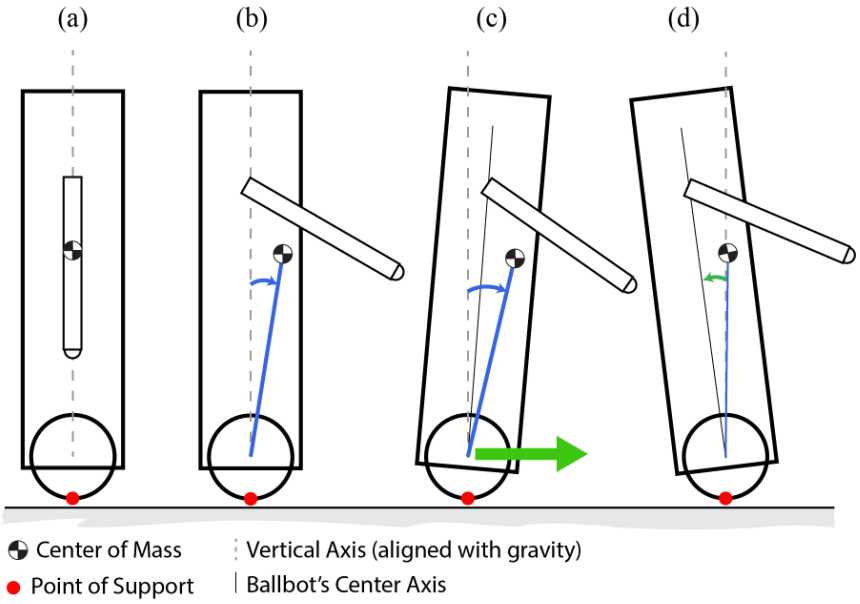


Figure 3.2: Sketch of depicting the effects of changes in shape configuration on the position variables.

3.1.3 Planning in Shape Space

The shape space trajectory planner developed by Nagarajan [40, 43] used dynamic constraint equations to plan trajectories in the shape space, which, when tracked, resulted in approximate tracking of desired position trajectories. The planner can handle systems with more shape variables than position variables and handles cases where a subset of the shape variables is artificially constrained. This framework was shown to experimentally work with the ballbot with 2-DOF arms with 2 kg weights attached to the distal ends. However, this planner required knowing the mass of the arm payload beforehand and could not adapt online to changes in payload mass. Moreover, extending this framework to the ballot with 7-DOF arms is nontrivial as the algorithm requires decomposing the 3D dynamics of the system, which for the 7-DOF system is a complex task due to the highly non-linear coupling terms. This framework plans arm trajectories to track a desired ball position trajectory indirectly. It does not plan for desired end-effector trajectories that are important for manipulation tasks.

3.1.4 Feed-forward COM Offsets Compensation

The controller presented by Vaidy et al. [35] compensates for shifts in the equilibrium point of the ballbot caused by small COM offsets in the body and sloped floors by adding feedforward body lean angle terms to the PD-PID cascading balancing controller. Two instances of this controller run simultaneously, one to control the motion of the system in the frontal plane, the other to control the system in the sagittal plane. The feedforward body lean angle compensation terms are computed using an online estimator based on the internal dynamics of a planar model of ballbot without arms. This model simplification assumes operation around a small range of body lean angles, *i.e.*, $-2^\circ < \phi < 2^\circ$. Manipulation tasks or handling heavy payloads break this condition. When there is a large change in the payload (*e.g.*, adding or removing a > 10 kg payload instantaneously) the estimator fails to detect the $> 3^\circ$ change in equilibrium point fast enough and may cause instability or undesirable jerk motions. Further, on average it takes the estimator 10 seconds to converge within 0.2° for small offsets of up to 1.35°. Instead, we replace the equilibrium body lean angle estimator with a kinematic-based closed-form calculation of the COM offset using direct measurements of the payload mass using the series-elastic actuators in the arm's shoulders. The proposed feedforward term calculation method in this thesis does not need a dynamic model. Consequently, it does not need to support the small-angle approximation, enabling a larger range of body lean angles, *i.e.*, $7^\circ > \phi$, for the controller to operate in.

3.1.5 Differentially Flat Path Planning

The COM offset strategy discussed in Sec. 3.1.4 requires an external planner to be able to navigate from point to point. In a similar fashion to the work in [35], the proposed framework in this thesis leverages the differentially flat path planner presented by Shomin [33]. The differentially flat path planner enables the ballbot to smoothly navigate autonomously through the environment while avoiding static and dynamic obstacles. The planner can generate dynamically feasible trajectories for the underactuated ballbot online. While the ballbot's acceleration is proportional to its body lean angle, its position depends on the ball angle θ. To enable a point-to-point motion

with an initial and final target condition, the *crackle* of the flat output is minimized, resulting in a ninth-degree polynomial trajectory for the ball path ($\theta_p(t)$) on the floor. Using the flat outputs and the 9th order polynomial, a feasible lean angle trajectory ($\phi_p(t)$) can be generated [33]. By directly tracking the body lean angle trajectory $\phi_p(t)$, the desired ball path $\theta_p(t)$ is indirectly tracked.

3.2 Approach

This section presents a control strategy to enable the ballbot to lift and transport heavy payloads. The strategy extends the balancing controller to compensate for the COM deviation due to the heavy load. We present a modification to the differentially flat-based planner to enable navigation while carrying a heavy load.

3.2.1 Balacing Controller with COM Compensation

The existing PD-PID cascading balancing controller for ballbot [17], discussed in section 2.3.1, was designed for the ballbot without arms. It also assumed that (1) the ballbot operated on a level floor, and (2) the COM of the robot was aligned with the center axis of the ballbot's cylindrical body (*i.e.*, there is no COM offset). Despite these assumptions, it has been shown to be robust against external disturbances and small COM offsets, and to work on the ballbot with 2-DOF arms [40].

Without modifications, manipulation tasks with heavy payloads (*e.g.*, carrying 10 kg payloads) result in large COM shifts that take the robot's state outside the range of attraction of the controller. The robot will accelerate until it becomes unstable and fall. To mitigate this we present an extension to the existing balancing controller by augmenting the desired body lean angle trajectory $\boldsymbol{\phi}_d(t, \boldsymbol{q}) \in \mathbb{R}^2$, as shown in Fig. 3.3. The objective of this controller is to enable underactuated systems, like the ballbot, to lift and carry heavy payloads while tracking a desired position trajectory $\boldsymbol{P}_s^d(t)$.

The body lean angle trajectory $\boldsymbol{\phi}_d$ is the net body lean angle necessary to balance the effects of (1) the navigation task, (2) manipulation task, and (3) model uncertainty. It is computed as the summation of three terms as:

$$\boldsymbol{\phi}_d = \boldsymbol{\phi}_m + \boldsymbol{\phi}_p + \boldsymbol{\phi}_{fb}. \tag{3.3}$$

The first term $\boldsymbol{\phi}_m(q) = [\phi_{m_x}, \phi_{m_y}]^T \in \mathbb{R}^2$ is a feed-forward body lean angle term to compensate for the effects of the manipulation task and payload mass. The term is computed from a closed-form expression based on the system's forward kinematics, detailed in Sec. 3.2.3. The generated body lean angle brings the COM back on top of the point of support. Intuitively, this approach shifts the old control inputs, at both inner and outer loop control levels, to operate about the actual equilibrium instead of a zero equilibrium.

The second term $\boldsymbol{\phi}_p(t) \in \mathbb{R}^2$ is the body lean angle trajectory, generated by a path planner, to track a desired position trajectory $\boldsymbol{P}_s^d(t)$. The term is generated from a modified version of the differentially flat path planner described in Sec. 3.1.5, the details of the modifications are presented in Sec. 3.2.4.

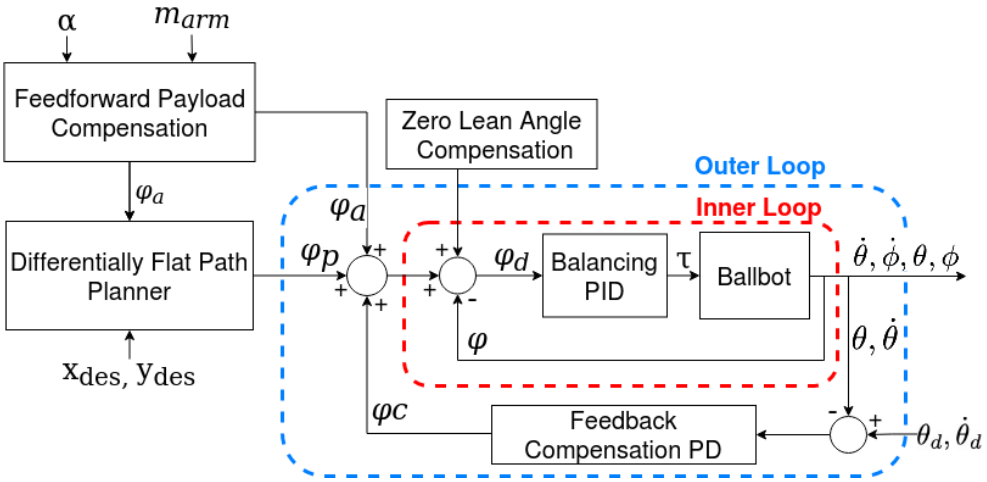


Figure 3.3: Overview of the implemented cascading control loops with feed-forward compensation terms.

Assuming we have a perfect robot and payload model, the previous two terms are sufficient to maintain balance and track the desired position trajectory. On real robots, this is rarely the case. There are unmodeled dynamics, nonlinear friction effects, noise, and mismatch in initial conditions. In order to solve these issues, the feedback compensation term $\boldsymbol{\phi}_{fb}(t, q) \in \mathbb{R}^2$ is introduced. This term is computed from a feedback position tracking controller that compensates for the deviation of the position trajectory from the desired trajectory as:

$$\boldsymbol{\phi}_{fb}(t, q) = K_p(q_p^d - q_p) + K_d(\dot{q}_p^d - \dot{q}_p), \tag{3.4}$$

where $K_p \in \mathbb{R}^{2\times2}$ and $K_d \in \mathbb{R}^{2\times2}$ are proportional and derivative diagonal gain matrices, respectively.

3.2.2 Mapping COM Position to Body Lean Angles

The Cartesian COM position of the ballbot with respect to the ball frame $\{S\}$ (*i.e.*, $\vec{\mathbf{G}}_{sys} = [G_{sys,x}, G_{sys,y}, G_{sys,z}]^T$) can also be represented as a rotation of the COM about the origin of frame $\{S\}$ by the COM offset angle $\boldsymbol{\phi}_G \in \mathbb{R}^2$, as shown in Fig. 3.4. The COM offset angle $\boldsymbol{\phi}_G$ is defined as the angle between the center axis of the ballbot and the axis connecting $\boldsymbol{P}_S$ with the system's COM. To maintain balance, the COM has to be brought back on top of the point of support. In the case of the ballbot, this is inducing a body lean angle $\boldsymbol{\phi}_{eq} \in \mathbb{R}^2$ to ensure that

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} - \begin{bmatrix} G_{sys}^x \\ G_{sys}^y \end{bmatrix} = 0. \tag{3.5}$$

The equilibrium lean angle is directly correlated to the COM offset angle, *i.e*, $\phi_{eq} = -\phi_G$.

When complete knowledge of the robot's and payload physical properties and state exist, then calculating $\boldsymbol{\phi}_{eq}$ is simple. Using forward kinematics the COM position of the entire system can

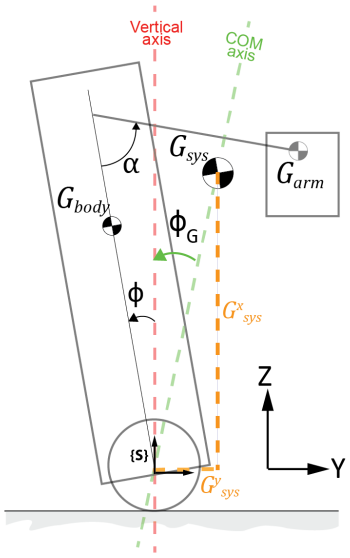


Figure 3.4: Schematic of the ballbot with 2-DOF arm COM angle calculation.

be computed. Knowing the new displaced position of the system's center of mass, the equilibrium body lean is calculated as:

$$\hat{\phi}_{eq} = -\text{atan}\frac{G^x_{sys}}{G^z_{sys}}. \tag{3.6}$$

In the task of lifting heavy boxes this is rarely the case. The mass of the box is usually unknown and can change through out the task execution.

3.2.3 Implementation with 2-DOF arms

When lifting heavy payloads with the ballbot with 2-DOF arms we make the following assumptions:

1. Lifting motions are only performed in the sagittal plane. The limited degrees of freedom of the arms restrict lifting motions with both arms to be in the sagittal plane. However, this simplifies the lifting problem into a 2D plane.
2. Both arms move synchronously and equally at all times. This ensures that both arms are sharing the load equally and the 2D projection holds.
3. The payload is always placed at the distal end of the arm links.
4. Full state information at any given time t_i is known, *i.e.*, the mass of the payload $m_{load}(t_i)$ and the state of the robot $q(t_i) \in \mathbb{R}^{8\times 1}$ is known.

If these assumptions hold, then a closed-form expression to compute $\hat{\phi}_{eq}$ can be obtained. The calculation is done for the projected system into the sagittal plane, as shown in Fig. 3.4 (b). The COM position of the system $\vec{\mathbf{G}}_{sys} \in \mathbb{R}^2$ with respect to the ball frame $\{S\}$ by

$$\vec{\mathbf{G}}_{sys} = \frac{\vec{\mathbf{G}}_{body} \cdot m_{body} + \vec{\mathbf{G}}_{arm} \cdot m_{arm}}{m_{body} + m_{arm}} \tag{3.7}$$

where $\vec{\mathbf{G}}_{body} = [G_{body,z}, G_{body,y}]^T$ is the COM of the cylindrical body, $\vec{\mathbf{G}}_{arm} = [G_{arm,z}, G_{arm,y}]^T$ is the COM of the arm with the payload, m_{body} is the mass of the cylindrical body, m_{arm} is the mass of the arms plus the mass of the payload. Note that because we make the assumption that the arm mass is negligible, $m_{arm} = m_{load}$. The value of the ballbot model parameters are listed in Table 3.1. The COM of the body $\vec{\mathbf{G}}_{body}$ is computed from:

Parameter	Value
m_{body}	81.65 kg
L_{com}	0.71 m
L_{joint}	1.3 m
L_{arm}	0.56 m

Table 3.1: Parameters of ballbot 2D model

$$\vec{\mathbf{G}}_{body} = -L_{com}\ \sin(\phi) \tag{3.8}$$

where L_{com} is the body center of mass along the z axis and ϕ is the ballbot's body lean angle with respect to the gravity vector. Following the assumption that the arm links are massless and the payload is always at the distal end of the arm, then $\vec{\mathbf{G}}_{arm}$ is computed from:

$$\vec{\mathbf{G}}_{arm} = L_{joint}\ \sin(\phi) - L_{arm}\ \sin(\phi - \alpha) \tag{3.9}$$

where L_{joint} is the shoulder joint height, L_{arm} is the arm length and α is the arm angle.

Knowing the new displaced position of the system's center of mass, the equilibrium body lean is calculated from Eq. 3.10 as

$$\hat{\phi}_{eq} = -\text{atan}\frac{G_{sys,z}}{G_{sys,y}}. \tag{3.10}$$

Combining Eq. 3.7 and Eq. 3.10 the required body lean angle $\hat{\phi}_{eq}$ to maintain balance is obtained as:

$$\hat{\phi}_{eq}(m_{load}, \alpha), = -\text{atan}\frac{m_{body}\, L_{com} + m_{load}(L_{joint} - L_{arm} \cdot \cos(\alpha))}{L_{arm} \cdot m_{load} \cdot \sin(\alpha)} \tag{3.11}$$

3.2.3.1 Mass Estimation

It is rarely the case that complete knowledge of the system properties and state exists. Usually, the mass of the payload is unknown or varying over time, resulting in an unknown COM position of the system. Leveraging the series-elastic actuators in the arms' joints, the robot can directly measure the mass of the payload online. Knowing the stiffness coefficient and deflection of the elastic element, the applied torque at the shoulder joint can be measured. By assuming that the arm link is rigid and the payload is at the distal end of the arm, we can estimate the payload weight from the definition of torque

$$\tau_\alpha = L_{arm} m_{load}\, g \sin(\alpha), \tag{3.12}$$

where τ_α is the torque magnitude at the shoulder joint, L_{arm} is the length of the arm (the position vector from the point about which the torque is being measured to the point where the force is applied), m_{load} is the mass of the payload, g is the acceleration due to gravity, and α is the angle between the gravity vector and the lever arm vector. Solving Eq. (3.12) for the mass of the payload results in

$$m_{load}(\tau_\alpha, \alpha, \phi) = \frac{\tau_\alpha}{L_{arm}\, g \sin(\alpha - \phi)}. \tag{3.13}$$

The series-elastic actuators on the 2-DOF arms joints are composed of a DC motor, a torsional spring, and two encoders at each end of the spring element. The torque at the shoulder joints can be measured directly from the series elastic actuators using Hooke's law:

$$\tau_a = -K_s \Delta\alpha, \tag{3.14}$$

where K_s is the spring constant and $\Delta\alpha$ is the deflection of the torsional spring. Combining Eq. 3.13 and Eq. 3.14 we measure the mass of the payload as a function of the shape variables $q_s = [\phi, \alpha]^T$ and the deflection of the torsional spring $\Delta\alpha$ as follows:

$$m_{load}(\phi, \alpha, \Delta\alpha) = \frac{K_s \Delta\alpha}{L_{arm} \cdot g \cdot \sin(\alpha - \phi)}. \tag{3.15}$$

Combining Eq. 3.11 and Eq. 3.15 a closed form solution for the equilibrium body angle in terms of the shape configuration is obtained:

$$\hat{\phi}_{eq}(\phi, \alpha), = -\text{atan}\left(\frac{m_{body}\, L_{com}\, g}{K_s \Delta\alpha} + \frac{L_{joint} - L_{arm}\cos(\alpha)}{L_{arm}\sin(\alpha - \phi)}\right). \tag{3.16}$$

This equation allows for fast computation of the equilibrium angle from direct physical measurements.

In experimentation, we noticed the spring did not act linearly, and the manufacturer's spring constant value resulted in inaccurate mass estimates. Instead, we performed a sensor calibration routine. The arms' joints were commanded to follow a sinusoidal trajectory while lifting payloads of different masses. The corresponding joint angle position and torque were recorded. Fig. 3.5 show the measured relation between payload mass m_{load}, arm angle α and joint torque τ_α. The X and Y-axis show the arm angle and the joint torque, while the Z-axis shows the correlating payload mass. The black cluster points correspond to a given payload mass. A quadratic relation was found that mapped τ_α, α to m_{load}, as follow

$$m_{load}(\tau, \alpha) = a\tau^2 + b\alpha^2 + c\tau\alpha + d\tau + e\alpha + f, \tag{3.17}$$

where the constants are $a = -7.357e^{-05}, b = 1.06, c = -0.05059, d = -0.2027, e = 0.906$, and $f = -4.841$. In the controller implementation Eq. 3.17 provides a better result than using Eq. 3.13.

3.2.4 Path Planning

The differentially flat formulation, described in Sec. 3.2.4, is based on the linearization of the planar ballbot without arms dynamics using a small-angle approximation. This model only holds

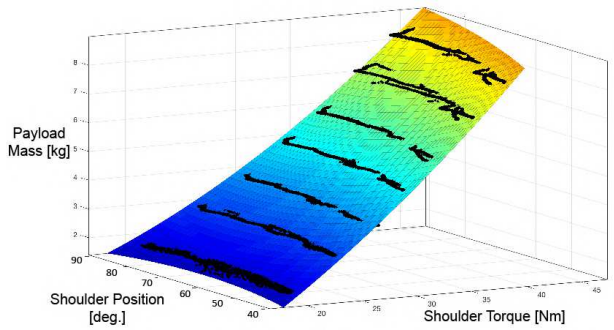


Figure 3.5: Fitted quadratic function to arm payload calibration data using Matlabs^{TM} curve fitting tool.

if the ballbot body lean angle is constraint to be between $-3° < \phi < 3°$. The COM angle offset $\hat{\phi}_{eq}$ when the ballbot is carrying a heavy object breaks this constraint. Consequently, the path planner cannot account for the induced body lean angle and generate trajectories that destabilize the ballbot. To mitigate this issue, instead of using the ballbot's state to plan trajectories, a virtual ballbot aligned with the COM axis was used, as shown in Fig. 3.6. In essence, this shifts the planner to be about the actual COM equilibrium instead of a zero body lean equilibrium point. To work, we assume that the arm and payload configuration does not change while the ballbot is navigating.

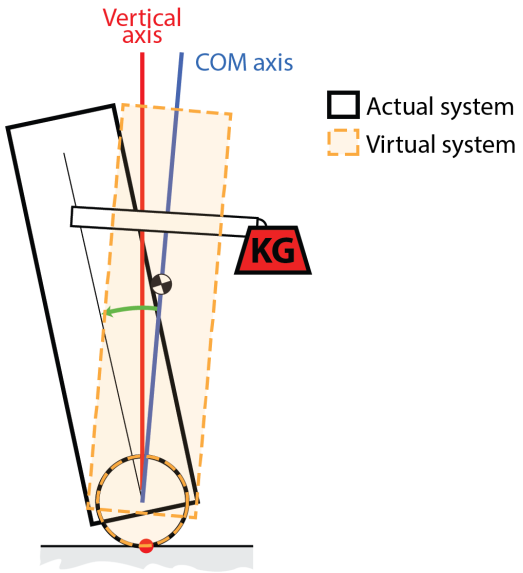


Figure 3.6: Schematic of virtual system (in orange) used for navigation with differential flat based planer while the real system (in black) is controlled with the COM compensation controller.

3.3 Experiments and Results

We experimentally validated the COM compensation controller presented in Sec. 3.2.3 and differential flatness planner with the modification shown in Sec. 3.2.4 on the ballbot with 2-DOF arms. To evaluate the proposed controller, the tasks of lifting and setting down payloads of unknown mass while station-keeping, transporting heavy loads from point to point, and transferring payloads with a human were considered. For all experiments, the payload mass was unknown for the robot initially. It actively measured the mass using the method described in Sec. 3.2.3.1. The results can be viewed in the supplemental Video C.1.

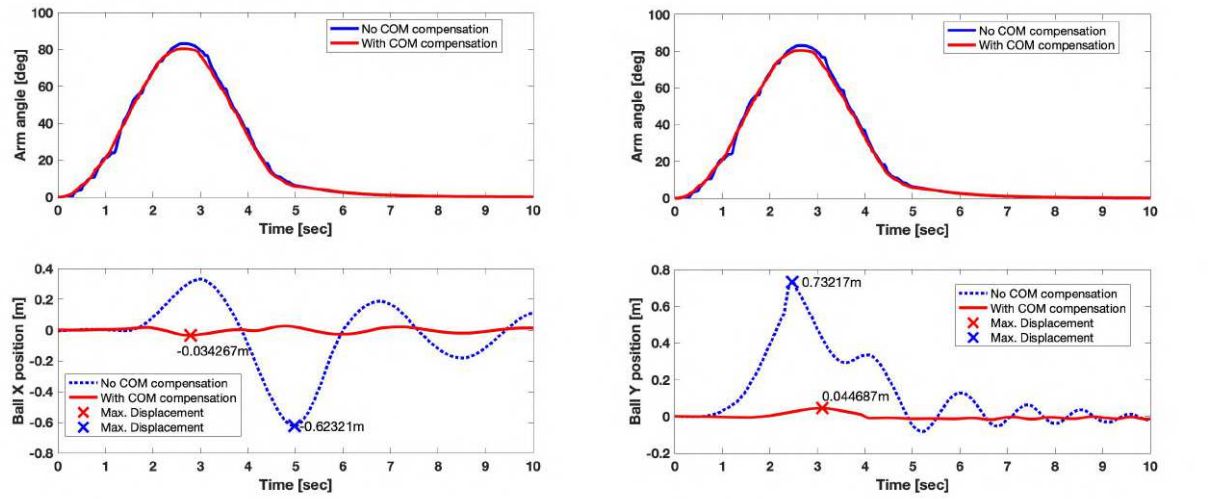
3.3.1 Station-Keeping while Lifting a Heavy Payload

tasks is to track the desired ball position. One specific case of this is *station-keeping*, *i.e.*, stay in one place $\dot{q}_p = 0$. The payload mass detection is constantly updating at a frequency of 500 Hz, enabling instantaneous changes in the payload. Fig. 3.7 shows the linear displacement of the ball while lifting a 10 kg heavy object with both arms. The actual lifting, where the arms were raised from 0° to 80°, occurred between second 0 and second 2.5. With the implemented COM compensator, the ball position error stayed within ± 0.045 m. This is a bigger tracking error than the ± 0.01 m error when the arms are not in motion or carrying a payload. However, once the robot reached steady-state the ball position was kept within ± 0.02 m. The increase in position error is expected as the ball must move to induce the required body lean angle to compensate for the load. Without the COM compensator, the ball position error was within ± 0.74 m. The implemented controller reduced the ball position tracking error by 86%. Fig. 3.8 shows screenshots of the robot performing this lifting task.

The maximum payload that the ballbot was able to lift was $m_{load} = 15$ kg. This is equivalent to the maximum theoretical payload capacity of the arms. Any larger payload mass would permanently deform the elastic element in the series-elastic actuators in the arms' joints. This is the first time the ballbot can lift this size payload. Fig. 3.9 shows the ballbot carrying a 15 kg heavy payload. To maintain balance, the ballbot leaned back 6.2°. Even with a 15 kg heavy object, the ballbot showed that it could yaw around its axis, at 10°/s.

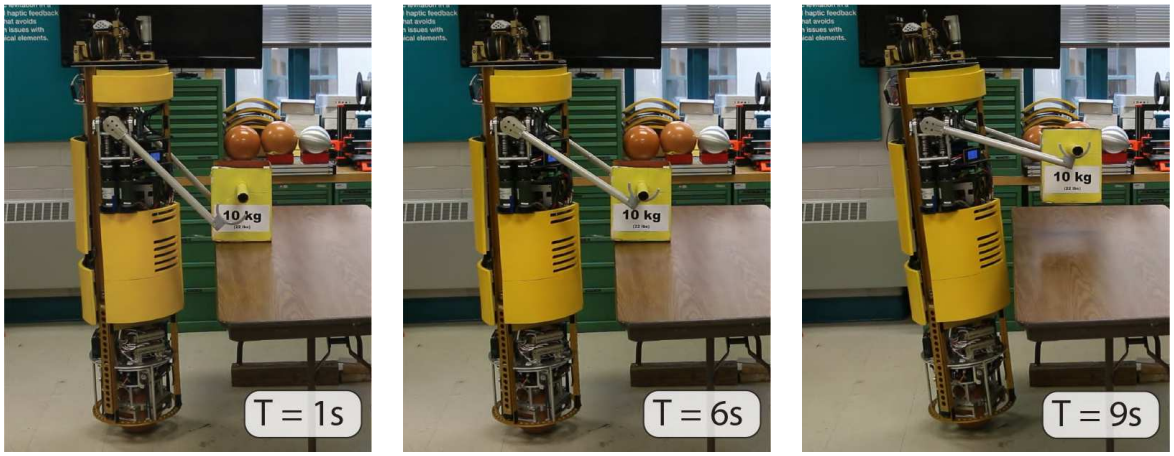
3.3.2 Payload Transport and Exchange with Humans

Another useful application for robots is transporting and exchanging heavy objects with humans safely and efficiently. The following experiments demonstrate the ballbot capable of realizing these types of tasks. Fig. 3.10 shows screenshots of the ballbot autonomously transporting a heavy payload and handing it over to a human. The ballbot carries a 10 kg payload, for which it has no prior knowledge, and transports it to a human 3 m away from it. The COM compensation controller described in Sec. 3.2.1 is actively measuring the payload mass and ensuring stability. It also tracks the desired body lean angle plan generated by the differentially flat planner described in Sec. 3.1.5. The controller can rapidly compensate for the sudden removal of the payload by the human, as shown in Fig. 3.10(c). The ball position stays within ± 0.02 m. The ballbot locomotes at 0.166 m/s while carrying the 10 kg payload.



(a) (b)

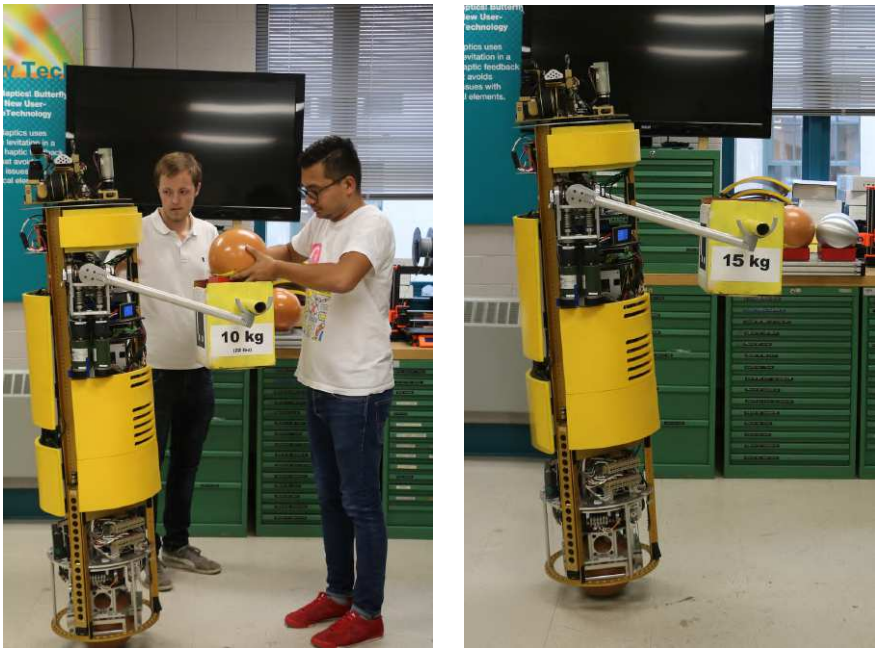
Figure 3.7: Ballbot motion while lifting a 10 kg payload, ball linear displacement in (a) the X axis, and (b) the Y axis. Arm joint angle is shown in top plots to correlate motion of ball with arm movement.



(a) (b) (c)

Figure 3.8: Screenshots of the ballbot lifting a 10 kg payload from a table while station-keeping. In screenshots (a) the ballbot is station-keeping, in (b) just before lifting the payload of unknown mass, and in (c) station-keeping while carrying the 10 kg payload.

The ballbot is also able to perform the complementary task. To navigate to the human, receive a payload of unknown mass, and navigate away. Fig. 3.11 shows screenshots of the ballbot receiving a 10 kg payload from a human, actively measuring the weight of the payload to compensate for it, and then navigates away with the payload. The estimated weight of the payload is within ± 0.5 kg. This is sufficiently accurate to compensate for its effects and track the desired ball trajectory. The observer in [35] is too slow to estimate the COM displacement in the



(a) (b)

Figure 3.9: The CMU ballbot station keeping using the COM compensation strategy while (a) receiving a payload mass and (b) carrying a 15 kg payload using its 2-DOF arms.

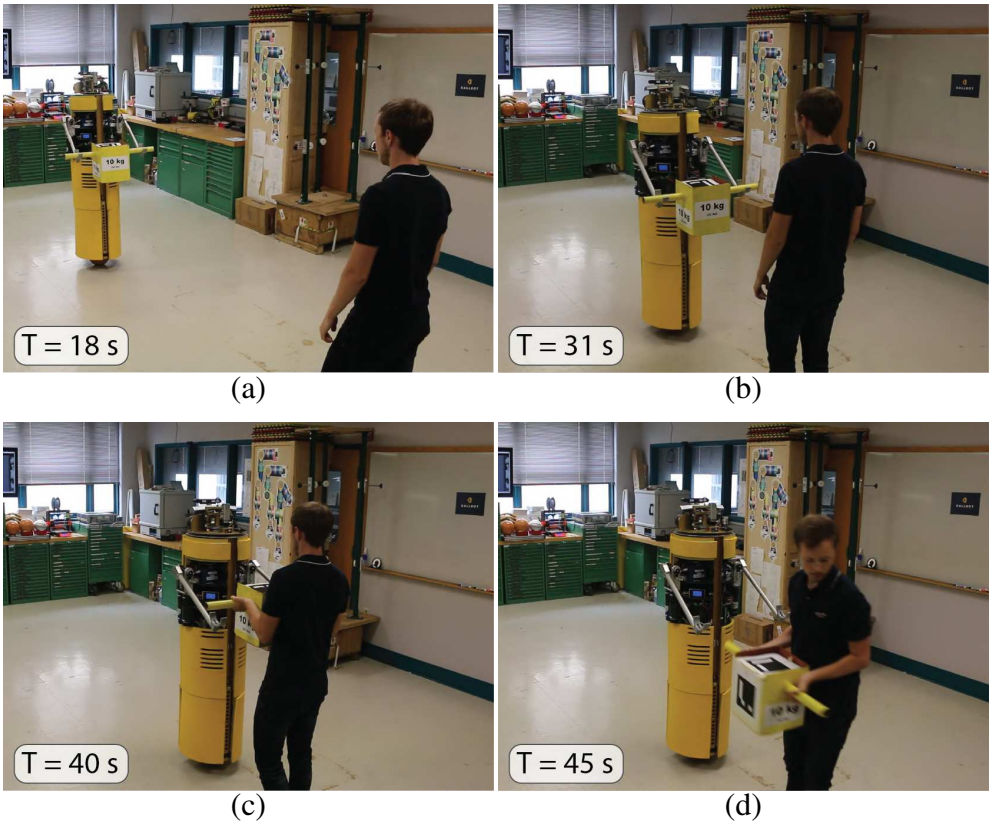
hand-off tasks, making it unfeasible for this experiment. The time between touching the robot's "hands" and the subject releasing their grasp was measured on average to be 5 s. It is half the time it takes the observer in [35] to converge to a COM estimate.

3.3.3 Lift and Place Task with in Place Yaw

This final experiment demonstrates the ballbot lifting a box from a table and placing it back at a different location in the table while station keeping. Fig. 3.12 shows screenshots of the ballbot lifting a box from a table, yawing $90°$, and setting it down on another section of the table. The ballbot is station-keeping in front of the table while it moves its arm from $0°$ to $80°$ to lift the box with a 10 kg payload. The lifting motion takes 8 s, and the ballbot is unaware of the payload mass. Once the ballbot satisfactorily lifts the payload, it is commanded to yaw $90°$ in place. Yawing while lifting a heavy payload is difficult because there is little rotational control authority over the contact point between the ball and floor. After facing the new table section, the arms are commanded down to set down the box. The ballbot can successfully station keep while performing this entire task.

3.4 Discussion

The presented extension to the balancing controller enabled the ballbot with 2-DOF arms to balance while carrying a heavy payload and track a ball position trajectory. The experimental results

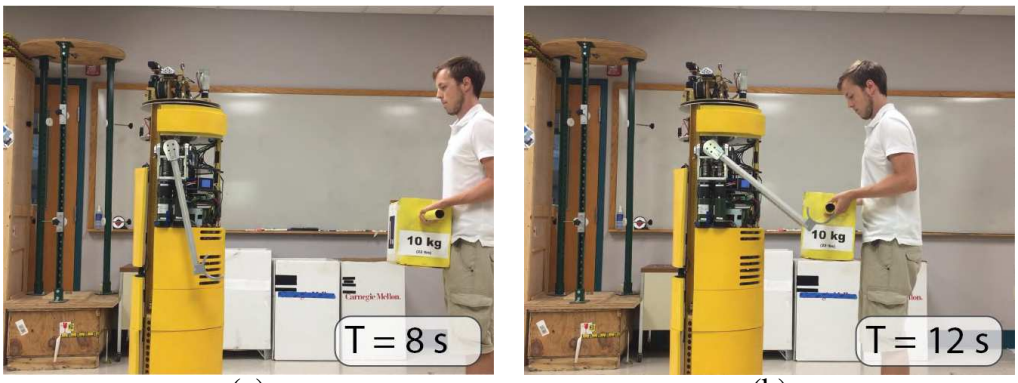


(a) (b) (c) (d)

Figure 3.10: Screenshots of the ballbot transporting a 10 kg payload over a 3.0 m distance and handing it over to a human. In screenshots (a) and (b) the ballbot is navigating autonomously, in (c) the human removes the payload, and in (d) the ballbot is station-keeping without the payload.

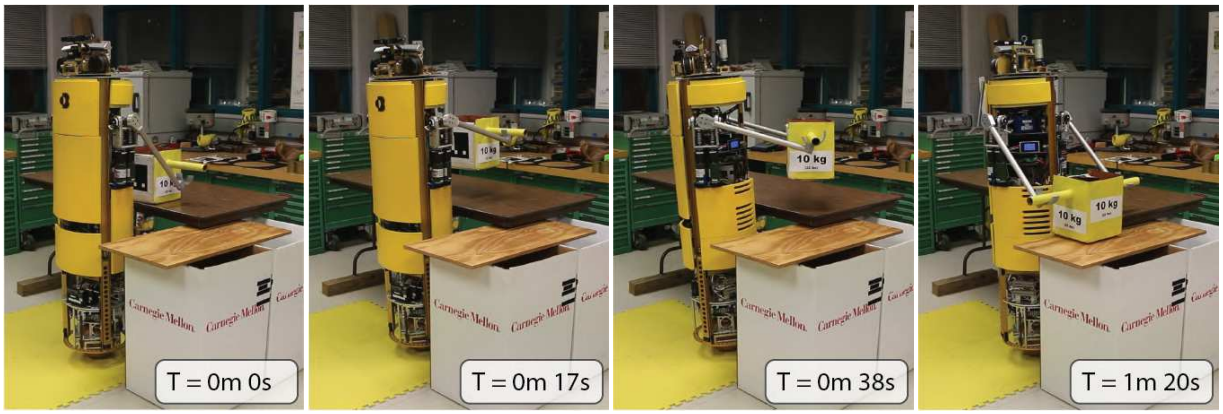
showed that the implemented differentially flat path planner can generate feasible locomotion plans when carrying heavy loads. However, it fails to consider the arms to perform combined manipulation and locomotion tasks. In the presented scheme, arm motion only happens when the ballbot is station-keeping. In Chapter 6, a whole-body planning framework based on simple centroidal dynamics and kinematics is introduced to overcome this limitation.

The manipulation experiments were restricted to simple motion due to the limited DOF. The ballbot was able to carry a maximum payload of 15 kg. The series elastic joints in the arms limited the lifting capacity. The ballbot could lift larger payloads using the same control strategy with stronger arms. Nevertheless, the experiments highlight the potential benefits of a mobile manipulator like the ballbot with 7-DOF arms. The human-robot and robot-human payload exchange tasks showed that the ballbot can collaborate close to humans in a safer manner. Also, they demonstrate that a direct measurement of the payload mass allows for dynamic weight adaptation, ensuring that the robot will remain stable despite sudden changes in the payload. The design of a new pair of 7-DOF arms is presented in the next chapter.



(a) (b) (c) (d)

Figure 3.11: Screenshots of the ballbot receiving a 10 kg payload from a human and navigating away to a predefined target. In screenshot (a) the human is approaching the robot, in (b) the human is handing over the payload of unknown mass, in (c) the ballbot leans back to compensate for the payload mass, and in (d) the ballbot navigates away with the payload.



(a) (b) (c) (d)

Figure 3.12: Screenshots of the ballbot lifting and placing a 10 kg payload while yawing in place. In screenshot (a) the ballbot is station keeping in front of the payload, in (b) the payload is picked up, in (c) the robot is midway through its yawing motion, and in (d) the ballbot is setting the payload down.

Chapter 4

Design and development of 7-DOF arms

The CMU ballbot equipped with a pair of simple 2-DoF arms [13], shown in Fig. 2.2 (b), has demonstrated to be a very capable and exciting robot that has particular relevance to the field of physical human-robot interaction. Sec. 2.1 summarized the manipulation capabilities of the ballbot with 2-DOF arms, ranging from guiding people by the hand [44] to aiding in sit-to-stand maneuvers [7]. However, the simple 2-DOF arms severely limit the range of manipulation tasks the ballbot can perform due to the lack of strength, dexterity, and large workspace.

This chapter presents a new high-performance multi-DOF pair of robotic arms for the CMU ballbot. The new arms increase the ballbot's manipulation workspace, payload capacity, and dexterity. The goal is to enable the enhanced ballbot to lift and carry up to 20 kg loads with both arms and perform more intricate manipulation tasks.

This chapter also demonstrates that with minor modifications, the proposed controller in Chapter 3 can be easily implemented on the ballbot with 7-DOF arms. Successful balancing while moving the arms is demonstrated in various experiments.

4.1 Background

In recent years many collaborative robotic manipulation platforms (cobots) intended to physically interact with humans in a shared workspace have been developed. The ABB YuMi [45], Yaskawa MOTOMAN SDA-Series [46], KAWADA NEXTAGE [47] are examples of available dual-arm systems. They can perform tasks such as loading, packing, and material handling. However, they either have low payload capacity (YuMi and NEXTAGE payload capacity ranges between 0.5 kg - 3.0 kg) or are mounted to large heavy fixed bases that make them impossible to be integrated into the CMU ballbot (MOTOMAN dual-arm system weighs 220 kg).

The KUKA LBR iiwa based on the DLR LWR III [48] and the UR10 by Universal Robots [49] are examples of robotic arms that have high positioning precision, rich proprioception that enables the robot arms to compliantly interact with the physical world, and a high payload capacity of 14 kg and 10 kg respectively. However, their usability for the enhanced ballbot is limited due to their heavyweights (LBR iiwa 29.9 kg and UR10 28.9 kg) and the need for a large separate control box that does not fit on-board. Further, the UR10 only has 6-DOF and a non-anthropomorphic kinematic configuration. This configuration was selected to optimize

workspace volume while mounted on a table. The 7-DOF Franka Emika Panda arm [50] and Kinova Gen3 Ultra lightweight [51] robotic arms are lighter alternatives. They weigh 17.8 kg and 8.2 kg but have a limited payload capacity of 3 kg and 4 kg, respectively. The Panda arm also requires a large 7 kg external control box. Thus, they are also not feasible options to be integrated into the CMU ballbot.

Few robotic arms have been developed in research facilities that achieve low weight and high payload capacities, such as the KIT dual-arm system [52] and the bi-manual manipulation platform by the Istituto Italiano di Tecnologia (IIT) [53]. The KIT arm consists of two 8-DOF, strong (11 kg payload), anthropomorphic robot arms. The IIT system is a pair of high-impact resistant 7-DOF robot arms with a superior weight to payload ratio (0.85 for continuous lifting operation). The IIT arm weighs only 8.5 kg and has a maximum payload capacity of 7 kg continuous and 15 kg peak. However, both manipulation systems are assembled from non-commercial custom sensor-control-actuator units, contrary to our developed arms. Further, the KIT dual arm is heavy and large (25 kg weight per arm and 1224 mm arm reach).

Unfortunately, none of these commercially available robotic arms meet the requirements of scale, weight, and actuation strength for the ballbot detailed in Sec. 4.2. Fig. 4.1 compares the payload to weight ratio of existing robot arms. It is desirable to have a large payload capacity and low weight (*i.e.* be close to the upper left corner). However, most are found in the bottom right corner of the plot where weight is large, and payload capacity is small.

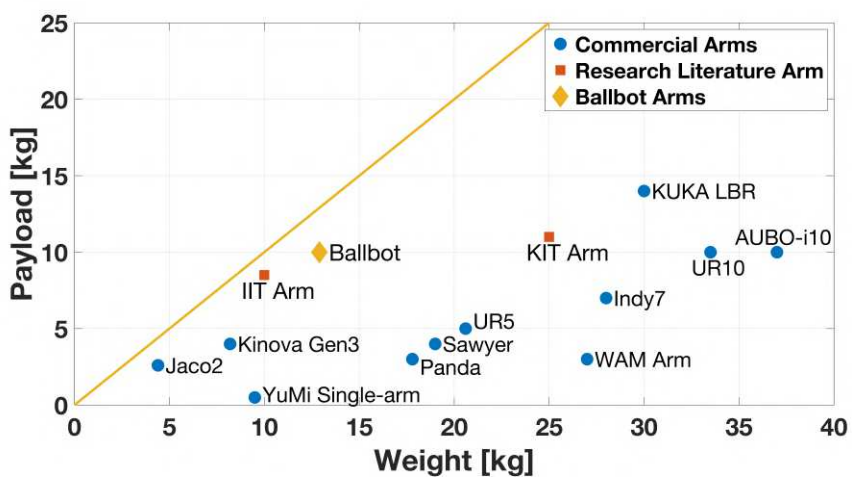


Figure 4.1: Plot of arm weight vs. payload capacity for different robotic arms. Commercial and research literature arms are shown.

4.2 System Requirements

The objective of this thesis work is to create an *agile and dexterous mobile manipulator* capable of being deployed in human-robot collaborative tasks and environments. With the long-term vision of deploying ballbot type robots in human environments, it is desirable to design arms suited

to operate in human environments. Human-robot collaborative tasks benefit from rich proprioception, large payload capacity, and physical resilience—leading to the following specifications:

1. **Arm Dimensions:** The ballbot was designed to be of human proportions. It is desirable to keep the arms in the same proportions (*i.e.*, average human arm is 0.608 m [54]), so they can be readily used in environments designed for humans. Further, they need to be lightweight to be mounted on the ballbot (*i.e.*, $\sim$ 10 kg per arm).
2. **Power Density:** The goal is to enable the enhanced ballbot to lift and carry up to 20 kg loads with both arms (*i.e.*, 10 kg payload per arm). Since the arm must be lightweight, a $\sim$1.0 weight-to-payload ratio is the target.
3. **Workspace:** A big limitation of the existing 2-DOF arms is the ability to manipulate small objects with both arms and hands. It is necessary to have a large bi-manual workspace in front of the robot (*i.e.*, human bi-manual workspace is $\sim$0.3 m^2 [54]), where both arms can interact with an object at the same time. Workspace is also related to the kinematic configuration of the system. It is necessary to have a similar kinematic structure as a human arm (*i.e.*, 7-DOF).
4. **Safety:** When operating around and with people, robots must embody soft, gentle compliant behavior. Correspondingly the arms must have passive and/or active compliance to react compliantly to unanticipated collisions. Recall the ballbot itself reacts with *intrinsic compliance* due to its balancing controller, which will add to any compliance of the arms.
5. **Physical Robustness:** The proposed manipulator platform is intended to be used for research purposes. Potential experiments include dynamic interaction with the environment that encompasses pushing off and colliding with walls. The robot arms should be able to withstand large perturbations and impacts. For extreme impacts, a mechanical safety mechanism should be implemented.
6. **Sensing and Proprioception:** Many novel algorithms today exploit the rich range of sensing modalities available. High-resolution position, inertial, and torque sensing are a minimum requirement. Additional vision and touch sensors at the wrist and hand are desirable. The ballbot platform already has a wide range of sensors onboard, complementing any new sensing capabilities of the arms.
7. **Power and Electronics:** The arms are intended to be interfaced with the existing CMU ballbot. Accordingly, they should be powered from the onboard 48 V batteries. They should also be self-contained with no external power or control box.

The specification listed above are summarized in Table 4.1. The following sections of this chapter describe the mechanical design decisions made to achieve the specified requirements.

4.3 Mechanical Design

An overview of the proposed arm components is illustrated in the CAD rendering in Fig. 4.2(a). The completed developed 7-DOF robot arms with human-like kinematic configuration integrated into the ballbot are shown in Fig. 4.2(b). Each arm has a maximum reachable distance of 815.5 mm without an end-effector (EE) and weighs 12.9 kg with a 10 kg payload capacity (0.78

Property	Functional Requirement	Specification
Size	High power and strength density	mass-to-payload ratio $\geq$1.0
	Lightweight to be compatible with ballbot	Mass $\sim$10 kg
	Arm dimensions and proportions comparable to those of human	Shoulder-Elbow dist.: 360 mm Forearm-Wrist dist.: 310 mm
Sensing	High proprioception	position, inertial and torque sensor on each joint
Usefulness	Lift and carry large payloads	10 kg payload per arm
	Large bi-manual workspace; similar to that of a human	$>$0.3 m^2
Robustness & Safety	Physical robustness against perturbations and impacts	Safety factor $\geq$1.5, use mechanical safety mechanism
	React compliantly to unanticipated disturbances passive or active compliance	active or passive compliance

Table 4.1: Summary of functional requirements and specifications for the 7-DOF arms.

payload-to-mass ratio). 49.1% (6.34 kg) of the total arm mass is considered static, only 6.56 kg are dynamic, reducing the required joint torques. When compared to thirteen of the most common robotic arms, the ballbot arms exhibit notable weight and strength, as shown in Fig. 4.1. Only 31% of the arms have equal or larger payload capacity than the ballbot's arms. The KIT Arm and KUKA LBR can carry 1 kg and 5 kg more than the ballbot's arms, respectively. However, their larger payload capacity comes with significantly greater weight. These arms weigh $>$93% more than the ballbot's arms. The UR10 and AUBO-i10 that have the same payload capacity as the ballbot's arms weigh $>$159% more.

Compliant arm behavior is achieved through a lightweight structure combined with high-resolution torque sensing [55]. A ball-detent torque limiter in the shoulder joint decouples the joint actuator from the affected links to ensure physical robustness to extreme impacts. The arms' major components and joint configuration are summarized in Fig. 4.2.

4.3.1 Arm Kinematics

The motions of a human arm can be equivalent to seven revolutions: shoulder abduction-adduction, shoulder flexion-extension, upper arm external-internal rotation, elbow flexion-extension, wrist abduction-adduction, wrist supination and wrist flexion-extension [9], as shown in Fig. 4.4 (a). With 7-DOF, each arm can manipulate all 6 DOF of the environment with adequate dexterity while using the additional DOF to resolve constraints introduced by the surrounding environment. There are many 7-DOF arm configurations. This thesis considered four different configurations, as shown in Fig.4.3 [9].

The ballbot arm kinematic configuration has a 3-DOF shoulder, 1-DOF off-center elbow, and 3-DOF non-spherical wrist, as shown in Fig. 4.4 (b) and (c). The off-center elbow was implemented to allow a greater elbow flexion range of motion. Similarly, a non-intersecting-axes wrist design was chosen to increase the wrist joint's range of motion. In contrast to other

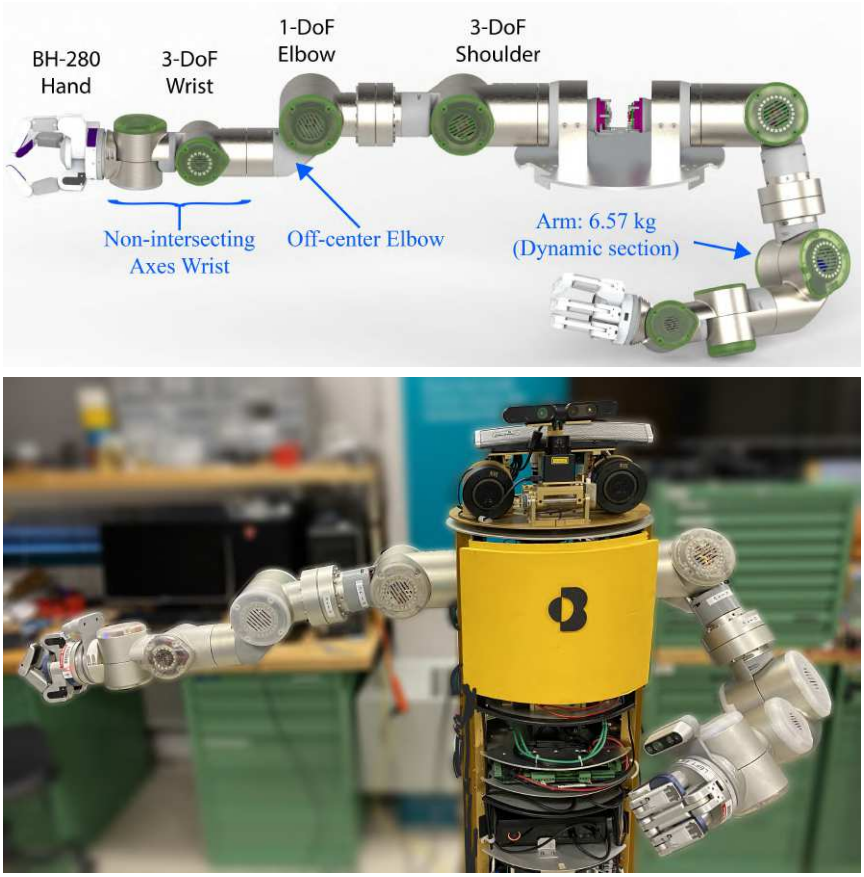


Figure 4.2: CMU ballbot dual arm system with BH-282 grippers, (top) CAD rendering and (bottom) arms interfaced with ballbot.

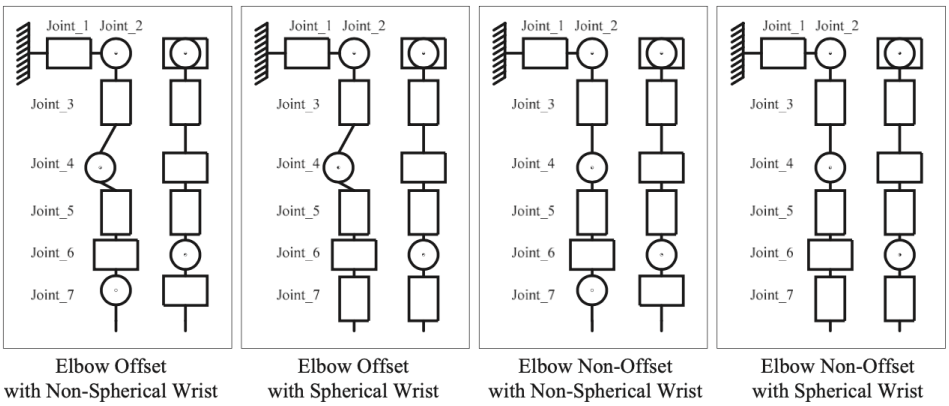


Figure 4.3: Typical 7-DOF arm configurations [9]

7-DOF arms, the ballbot's arms are of similar scale to that of an average adult human. Table 4.2 shows that the ballbot arm's dimensions are closest to that of a human arm in comparison to other robotic arms with similar payload capacity. From shoulder to end-effector tip, the arm measures

735.5 mm, only 20 mm more than a human arm (710 mm [56]). Note that at full extension, from the arm base to the EE tip, the arm has a reach of 935.5 mm, but a 200 mm shoulder section is located inside the ballbot's body and reduces the effective reach to 735.5 mm. The complete arm dimensions are summarized in Fig. 4.5.

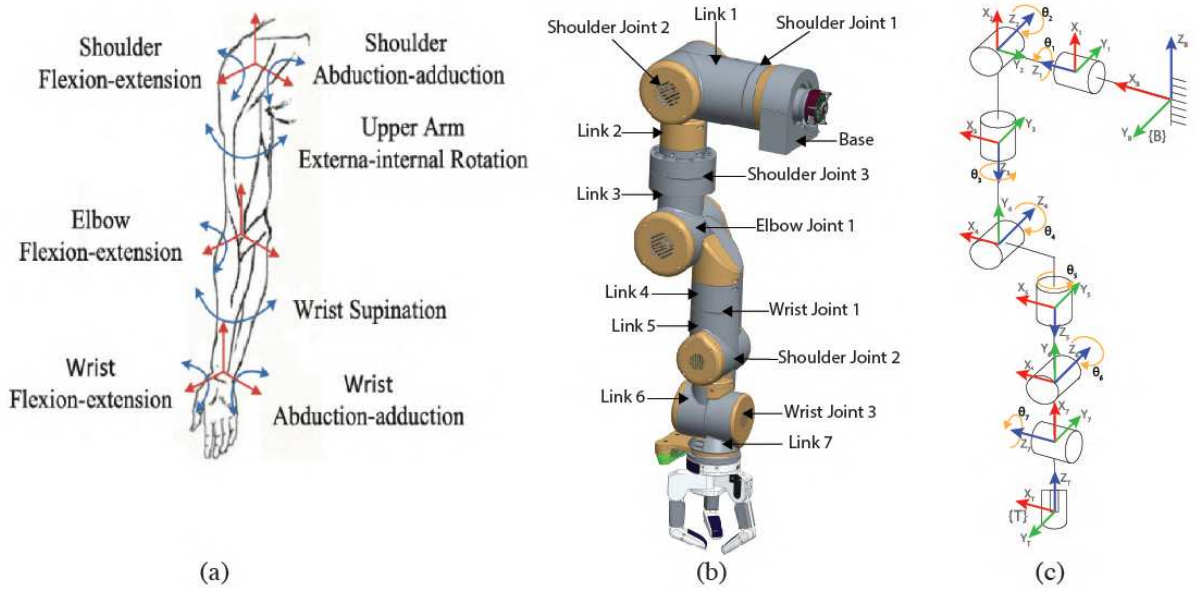


Figure 4.4: Overview of (a) the degrees of freedom of a human arm [9], (b) the joints and links of the 7-DOF arm, and (c) the kinematic tree and frames of the 7-DOF arm.

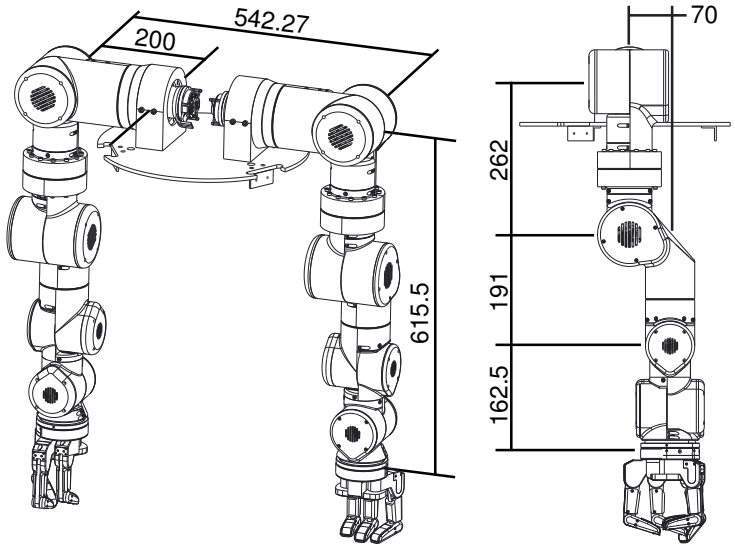


Figure 4.5: Size specification for pair of arms, all dimensions in mm.

Distance	Human [56] [mm]	KIT [52] [mm]	IIT [53] [mm]	KUKA iiwa [57] [mm]	Ballbot [mm]
Shoulder - Elbow	303	409	293	400	262
Elbow - Wrist	270	364	410	526	353.5
Wrist - EE Tip	137	227	–	–	120
Shoulder - EE Tip	710	1000	(703)	(926)	735.5

Table 4.2: Human arm vs. 7-DOF robot arms dimensions.

4.3.2 Range of Motion and Workspace

The target joint range and workspace were defined considering human ergonomic data [56]. The human arm range of motion was used as a starting point, but when possible, a greater range was implemented to enhance the motion and manipulation capability of the arm. The ranges of the upper and lower shoulder rotations (Joint 1 and Joint 3) were extended to be capable of full $360°$ rotation. This maintains the ballbot symmetry about the frontal plane, allowing to effortlessly switch between a front and back bimanual workspace in tight locomotion spaces without rotating the body. The range of the elbow (Joint 4) was increased by implementing an off-center elbow joint arrangement that results in a $30°$ extension and $160°$ flexion, as shown in Fig. 4.6. The

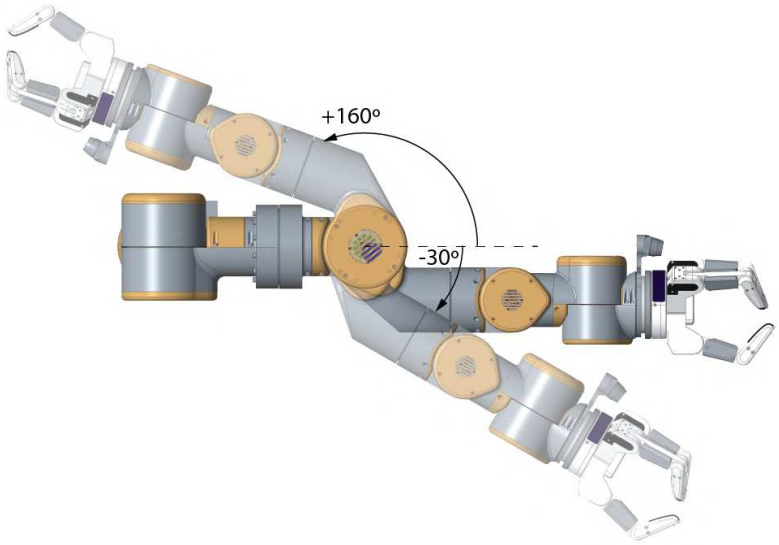


Figure 4.6: Range of motion of off-center elbow

wrist joints are arranged in a non-anthropomorphic configuration with non-intersecting axes to maximize the range of motion of the wrist flexion/extension and abduction/adduction (Joint 6 and Joint 7) motions to $[-90°, 90°]$, as shown in Fig. 4.7. The range of motion of all joints is summarized in Table 4.3. The workspace of each arm is composed of a large hemisphere volume of radius 615.5 mm. This is an increase from the 580 mm radius hemisphere shell workspace of the 2-DOF arms. The increased range of motion of the joints foresaid allows for a greater

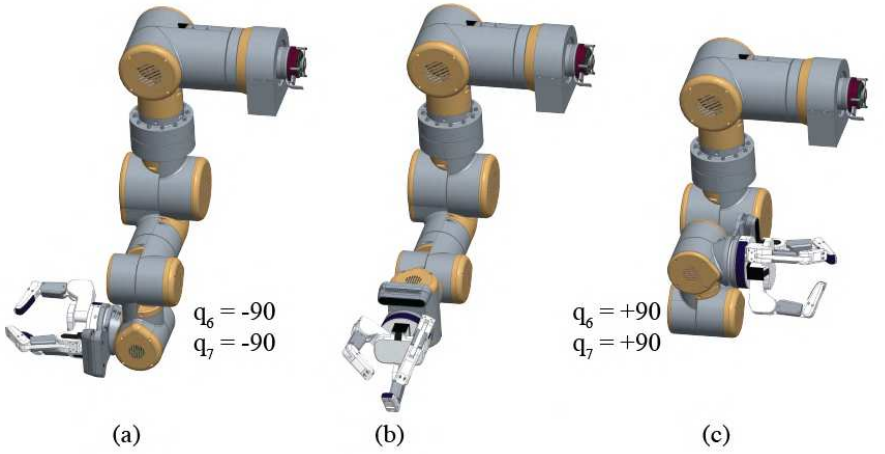


Figure 4.7: Range of motion of the non-spherical wrist

bimanual workspace of 0.46 m^2, compared to 0.3 m^2 of the average human [56], as shown in Fig. 4.8. The 3D workspace volume of both arms is depicted in Fig. 4.9. Note that the entire three-dimensional range of motion in front and behind the robot is accessible to the arms.

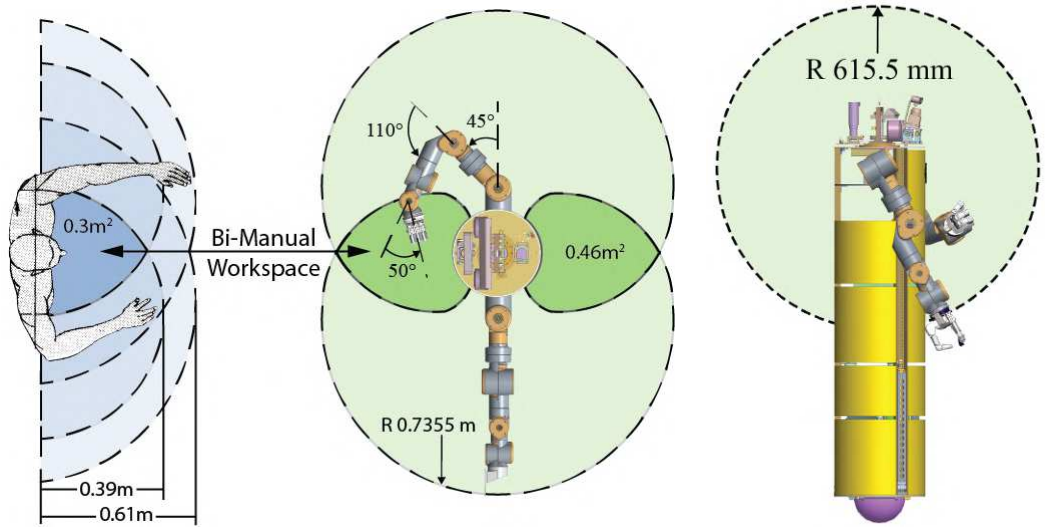


Figure 4.8: The 2D workspace of the ballbot arms; (a) top view compared to human arm workspace, dark green represents the bi-manual workspace area; (b) side view of a single arm workspace.

4.3.3 Actuator and motor driver selection

To determine the actuation requirements, a dynamic model of the 7-DOF arm was developed using MATLAB's Robotics Toolbox [58]. Inertia and mass properties were estimated from a

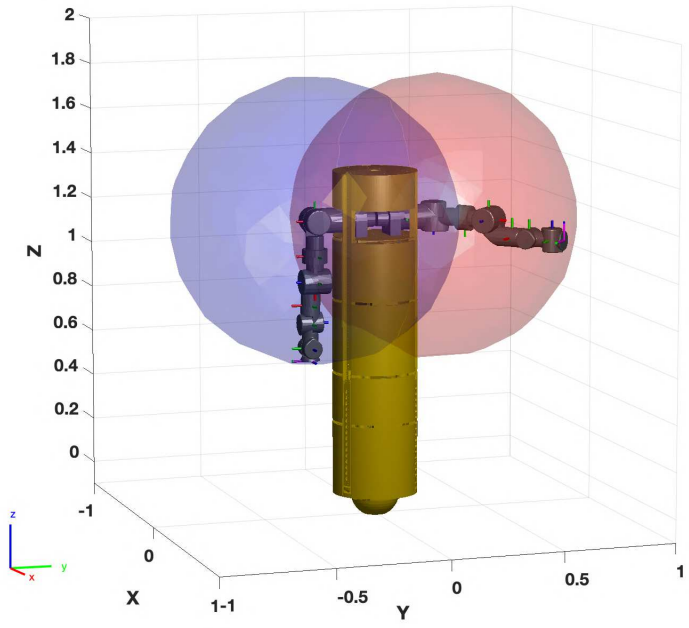


Figure 4.9: The 3D workspace of both arms, blue volume corresponds to the left arm and the red volume to the right arm. The intersection of both volumes is the dual arm workspace.

Table 4.3: Mechanical Properties of the 7-DOF Arm

Joint No.	Articulation	Range [deg.]	Actuator Type [SENSO-Joint]	Gear Ratio	Torque [N/m] peak - nominal	Max. Velocity [rpm]	Mass [kg]
1	Shoulder flexion/extension	[-720, 720]	100 RD5014 AEST	160	120 - 56	21	1.45
2	Shoulder abduction/adduction	[-10,190]	100 RD5014 AEST	160	120 - 56	21	1.45
3	Shoulder rotation int./ext.	[-720, 720]	100 RD5008 AEST	160	100.8 - 30	34	1.35
4	Elbow flexion/extension	[-30,160]	100 RD5008 AEST	160	100.8 - 30	34	1.35
5	Wrist rotation	[-720,720]	75 RD3806 AEST	100	19 - 5.4	85	0.7
6	Wrist flexion/extension	[-90,90]	75 RD3806 AEST	100	19 - 5.4	85	0.7
7	Wrist abduction/adduction	[-90,90]	75 RD3806 AEST	100	19 - 5.4	85	0.7

CAD model of an initial prototype of the 7-DOF arms. A set of 30 different joint trajectories that covered the entire workspace were simulated with a 10 kg payload mass attached to the end of the arm and subject to gravity. The recursive Newton-Euler algorithm was used to solve the inverse dynamics problems and obtain the required joint torques. The box plot in Fig. 4.10 shows the torque range requirement by each joint to follow the 30 simulated trajectories. As expected, the shoulder joints demand the highest torque (105 Nm peak). From Fig. 4.10, it is apparent that joint torque requirements can be satisfied by using only three different actuator sizes with peak torques of 105 Nm, 60 Nm, and 25 Nm.

Each arm is built around three different custom sensor-actuator units by SENSODRIVE GmbH[1], shown in Fig. 4.11. The units used, *SENSO-Joint*: *100 RD5014 AEST*, *100 RD5008 AEST*, and *75 RD3806 AEST*, achieve peak torques of 120 Nm, 100.8 Nm, and 19 Nm, respec-

[1]SENSODRIVE GmbH: https://www.sensodrive.de

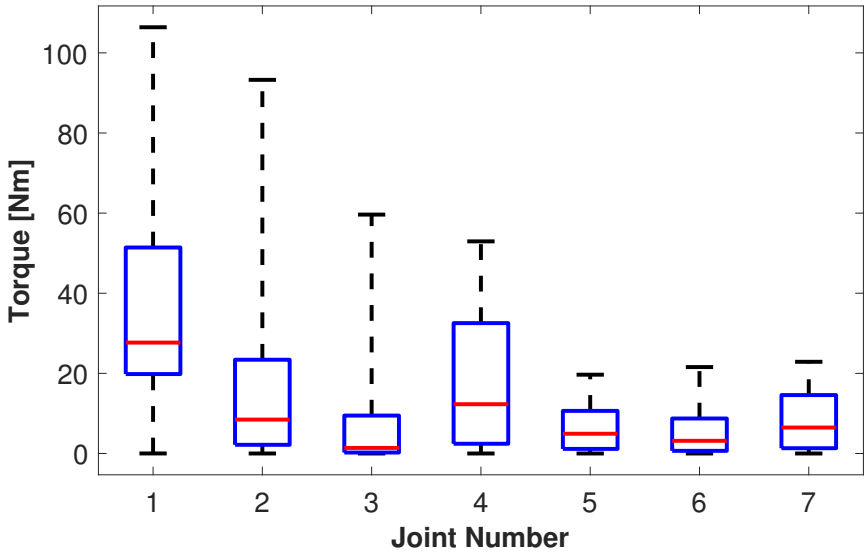


Figure 4.10: Box plot of joint torques recorded from the simulated trajectories moving a 10 kg payload. The black "whiskers" show the maximum and minimum torques, the average recorded torque is shown by the red line, and interquartile range by the blue box.

tively. The sensor-actuator units were developed in collaboration with SENSODRIVE for this project but are now commercially available[2]. Their complete specifications are summarized in Table 4.3. Each drive unit combines a TQ-RoboDrive BLDC motor, Harmonic Drive, cross-roller ring bearing that decouples the input and output, incremental and absolute encoders, motor temperature sensor, and output torque sensor in a compact, lightweight package. Detail specifications of each sensor resolution and accuracy can be found in Table A.1. A custom 50 mm diameter circular shape motor controller board, developed by MUSE Robotics[3], was integrated into the back of each SENSO-Joint, eliminating the need for a large external control box. The motor controller board includes the driver electronics, 6-axis IMU sensor, and communication and sensor interfaces. The only connections from each sensor-actuator-control unit are the DC-bus for power supply and the Ethernet bus for communication. Those connections are daisy-chained between units, and the cables are routed through the actuator's hollow shaft. This allows for $\pm 720°$ joint rotation, minimum cables, and overall smaller arm volume.

4.3.4 Arm Structure

The arm structure that links all the SENSO-Joint units is based on an exoskeleton approach using lightweight shell structures that are both load-bearing and covers [52, 53]. The actuators are floating inside the exoskeleton structure and are mechanically fixed to the link structure via screw flanges, as illustrated in Fig. 4.12.

The exoskeleton structure has several advantages over the classic frame construction. The bending stiffness K of a link is proportional to the material's Young's modulus E times the area

[2] https://www.sensodrive.de/news/cooperation-with-carnegie-mellon-university-cmu
[3] http://www.muserobotics.com/

Figure 4.11: SENSO-Joint sensor-actuator units from SENSODRIVE assembled with custom BLDC motor controllers by MUSE Robotics used in the ballbot's 7-DoF arm.

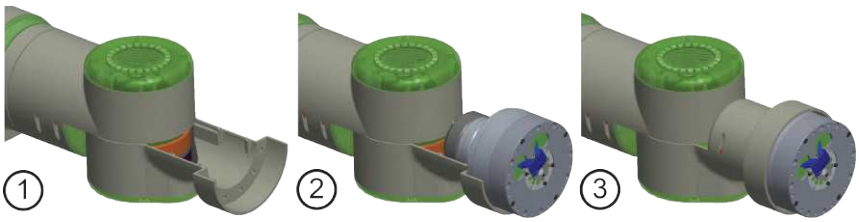


Figure 4.12: Integration steps of a sensor-actuator-control-unit into the exoskeleton shell structure

moment of inertia I and inversely proportional to its length L, *i.e.*, $K = 3EI/L^3$. Thus, the arm links can be optimized to be short hollow structures with large I with minimum openings that are lightweight while stiff to increase position precision.

The arm is assembled with only a few complex parts, in contrast to using many simple parts in traditional frame constructions. Loading torques and forces are only transmitted through the link's mounting flanges allowing thin wall structures with thicker flanges where loads are larger. The 3 mm wall thickness hollow exoskeleton links, shown in Fig. 4.13, were machined from single blocks of 7075-T6 aluminum using a 5-axis CNC milling machine. The links weigh between 0.242 kg - 0.454 kg. Some none-load-bearing covers were manufactured using 3D Printing.

4.3.5 End-Effectors

For dexterous manipulation, a three-finger Barrett Hand BH-280 [59] is attached to each arm. This robotic end-effector is a highly integrated, reliable, and fully-supported product that is in use in many robotics research programs from which future work can build on. Each arm has internal wiring to provide CAN communication and a 24 V power supply to the robotic hand. A USB 3.0 cable also runs inside the arm to provide a serial communication interface. The BH-280 provides finger joint torque-sensing, adding to the robot's sensing capabilities. Additionally, an

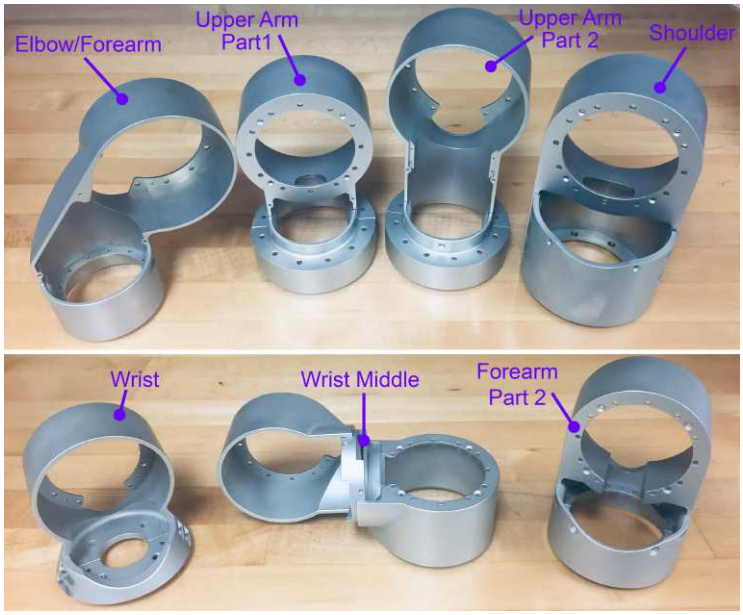


Figure 4.13: Exoskeleton shell structural links. Each link is machined from a single block of 7075 aluminum using a 5-Axis CNC milling machine.

Intel RealSense D435i is mounted at the wrist of each arm, to provide visual feedback for visual servoing. The arm wrist design allows for easy mounting of different grippers, hands, and tools to fit the task's needs. In addition to the BH-280, we have designed two passive end-effectors. The paddle hand was used for initial cuboid picking experiments, and the knob hand was used for pushing on walls experiments. Both passive end-effectors are 3D printed in PLA plastic and have a neoprene rubber surface to increase the contact surface friction. Students at Carnegie Mellon University have already made the most out of mounting and communication options to interface a novel 3D printed tendon driven soft hand [10]. Fig. 4.14 shows the four different end-effectors integrated to the ballbot.

4.4 Arm Controllers and Estimation

This section presents the implemented controllers and state estimation algorithms necessary to operate the new 7-DOF arms. We designed the controllers shown here, considering only the kinematics and dynamics of the new robot arms. We did not consider the kinematics and dynamics of the entire ballbot system. These controllers are the basis for developing more complex algorithms considering the complete ballbot manipulation system.

4.4.1 Joint Torque-Compliant Control

An important goal is to ensure compliant arm behavior, especially when interacting with people. Inspired by the results with DLR lightweight arm, which can react quickly to collisions and respond safely around humans [55], we follow the same approach to use a lightweight structure

(a) (b)

(c) (d)

Figure 4.14: Hands used with the ballbot arms, (a) the BH-280 hand used for active grasping tasks, (b) the custom low-cost dexterous soft hand [10] to explore compliant and soft interactions, (c) the paddle hand used for initial cuboid picking experiments, and (d) the knob hands for force experiments.

combined with high-resolution torque sensing to generate sufficient active compliance.

The implemented torque-compliant based feedback controller on each arm includes feedforward gravity and torque-sensing compensation, as shown in Eq. 4.1.

$$\boldsymbol{\tau}_d = K_{P_\alpha}(\boldsymbol{\alpha}_d - \boldsymbol{\alpha}) + K_{D_\alpha}(\dot{\boldsymbol{\alpha}}_d - \dot{\boldsymbol{\alpha}}) + g(\boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}) - \boldsymbol{\tau}, \tag{4.1}$$

where $\boldsymbol{\tau_d} \in \mathbb{R}^7$ is the vector of joint torque input commands, $\boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}} \in \mathbb{R}^7$ are the arm joint positions and velocities, $\boldsymbol{\alpha}_d, \dot{\boldsymbol{\alpha}}_d \in \mathbb{R}^7$ are the desired joint positions and velocities, $K_{P_\alpha}, K_{D_\alpha}$ are positive definite diagonal stiffness and damping gain matrices, $g(\boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}})$ is the gravity compensation term based on the dynamic model of the arm, and $\boldsymbol{\tau} \in \mathbb{R}^7$ is the vector of measured joint torques. An overview of the complete control loop is illustrated in Fig. 4.15.

The joint control was implemented on an Intel Core I7 @ 2.6GHz, running Ubuntu 16.04 Linux. The computer calculates motor current commands and sends them to each motor driver board via an Ethernet bus running at 250 Hz with a 100 Mbps data rate. The inner PI current control loop is executed on each motor driver at 10 kHz. To ensure that no mechanical joint limits are violated during operation, a unidirectional stiffening PD controller is implemented in the "Safety Check" block in Fig. 4.15.

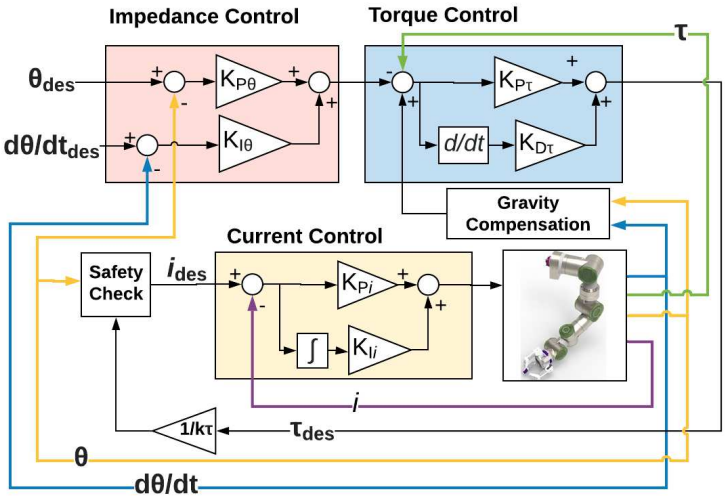


Figure 4.15: Block diagram of the impedance joint controller with gravity compensation. The motor current i is measured by the motor driver board, the joint torque τ is measured by a torque sensor and the motor position θ by an absolute encoder. The derivatives are obtained by numerical differentiation. Sensors' specifications are detailed in Table A.1.

4.4.2 Task-Space Admittance Control

Another important capability for the arms is to control its end-effector position. Admittance control is a popular strategy used for human-robot interaction. Admittance is the opposite of impedance. It takes guiding forces/torques by the human and maps them to task-space velocities. Task-space admittance control renders a virtual behavior for the end-effector that follows the dynamic equation

$$M_d\ddot{X} + D_d\dot{X} + K_sX = F_{virt} + F_{ext}, \tag{4.2}$$

where $\ddot{X}, \dot{X}, X \in \mathbb{R}^6$ are the Cartesian acceleration, velocity and position of the end-effector. The gains $M_d \in \mathbb{R}^6$, $D_d \in \mathbb{R}^{6\times6}$, and $K_s \in \mathbb{R}^{6\times6}$ are positive definite matrices describing the desired virtual inertia, damping and stiffness of the end-effector. $F_{ext} \in \mathbb{R}^6$ is the vector of measured external forces $f_{ext} \in \mathbb{R}^3$ and torques $\tau_{ext} \in \mathbb{R}^3$, *i.e.*, $F_{ext} = [f_{ext}, \tau_{ext}]^T$. $F_{virt} = [f_{virt}, \tau_{virt}]^T \in \mathbb{R}^6$ is a vector of desired virtual forces to modify the behavior of the end-effector dynamics. This control strategy is summarized in Fig. 4.16.

The desired Cartesian acceleration $\ddot{X}_{ref} \in \mathbb{R}^6$ necessary to follow the dynamic behavior described by Eq. 4.2 is calculated by

$$\ddot{X}_{ref} = M_d^{-1}[-D_d\dot{X} - K_sX + F_{virt} + F_{ext}]. \tag{4.3}$$

Using the Euler integration method, the joint the Cartesian end-effector reference velocity $\dot{X}_{ref} \in \mathbb{R}^6$ is obtained. The Cartesian velocities cannot be directly commanded to the robot arms. Thus, $\dot{X}_{ref}$ is mapped to reference robot joint velocities $\dot{q}_{ref} \in \mathbb{R}^n$ by the inverse differential kinematics

$$\dot{q}_{ref} = J(q)^\dagger\dot{X}_{ref}, \tag{4.4}$$

where $J(q)^\dagger \in \mathbb{R}^{n\times6}$ is the pseudo-inverse of the manipulator Jacobian,

$$J(q) = \frac{\delta X}{\delta q} \in \mathbb{R}^{6\times n}. \tag{4.5}$$

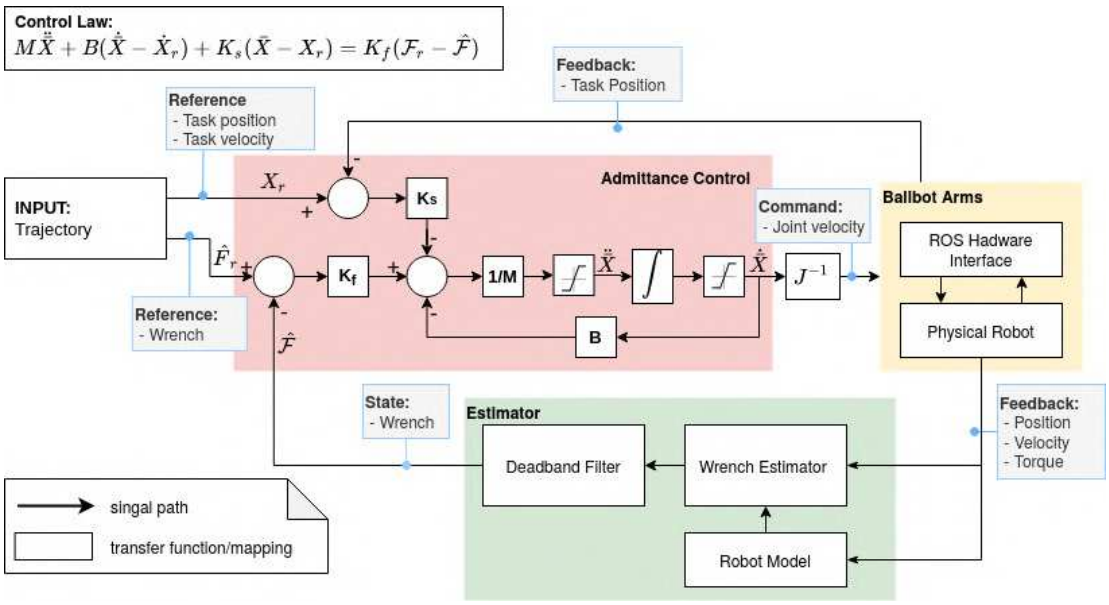


Figure 4.16: Block diagram of implemented Task-Space admittance controller on the ballbot arms.

The reference joint velocities are integrated using Euler integration to obtain reference joint positions $q_{ref} \in \mathbb{R}^{n\times 1}$. For the ballbot, $n = 7$ corresponding to the seven degrees of freedom of each arm. The reference joint position and velocities $q_{ref}, \dot{q}_{ref}$ are passed as input commands to a low-level position control system. For the ballbot arms, the external force F_{ext} is estimated from the joint torque sensors as described in Sec. 4.4.3. Note that this controller only controls the DOF of a single arm. Two instances of the controllers are implemented to control both arms. Further, this controller assumes that arms are mounted to a fixed base rather than a mobile base.

The formulation above is designed for regulating the relationship between position and force about a fixed operating end-effector pose. It is desirable to track an end-effector trajectory in some scenarios instead of just regulating the end-effector position. The control law can be extended by introducing the end-effector pose error dynamics to track a trajectory. Let $e = X_{ref} - X_{des}$ be the end-effector Cartesian pose error and $\dot{e} = \dot{X} - \dot{X}_{des}$ the Cartesian velocity error. Replace this trajectory tracking error terms in Eq. 4.3 as

$$\ddot{X}_{ref} = M_d^{-1}[-D_d\dot{e} - K_s e + F_{virt} + F_{ext}], \tag{4.6}$$

to obtain a reference Cartesian acceleration. The joint position and velocity commands are obtained the same way as before. The Cartesian position errors are composed of linear and angular components:

$$e = \begin{bmatrix} e_x \\ e_\theta \end{bmatrix} \begin{matrix} \in \mathbb{R}^{3\times 1}, \text{linear error} \\ \in \mathbb{R}^{3\times 1}, \text{angular error} \end{matrix} \tag{4.7}$$

The linear errors e_x are computed using Euler distance. The angular component error [60] is defined as

$$e_\theta = R_{ee}\epsilon_d^e \in \mathbb{R}^{3\times 1} \tag{4.8}$$

where $R_{ee} \in \mathbb{R}^{3\times3}$ is the rotation matrix for the end-effector, and ϵ_d^e is the vector part of the unit quaternion extracted from R_d^e. Which is defined as,

$$R_d^e = R_{ee}^\top R_d, \tag{4.9}$$

the error between the desired end-effector rotation $R_d \in \mathbb{R}^{3\times3}$ and the actual rotation R_{ee}.

4.4.3 End-Effector Wrench Estimation

The joint torque measurements are sufficient for joint-space controllers. However, for task space controllers, it is necessary to measure the wrench acting in the task space frame. A wrench estimator based on joint torque measurements was implemented instead of adding a 6-DOF Force/Torque sensor to the robot's wrist.

Consider the following Euler-Lagrangian dynamics of one of the robot arms

$$M(q)\ddot{q} + c(q,\dot{q}) + g(q) = \tau_m - \tau_{ext} \tag{4.10}$$

where $M(q) \in \mathrm{R}^{7\times7}$ is the inertia matrix, $C(q) \in \mathrm{R}^{7\times7}$ is the matrix containing the Coriolis and centrifugal terms, $g(q)$ is the gravity vector, τ_m is the joint torque, and τ_{ext} is the joint torque associated to an external force F_{ext} by

$$\tau_{ext} = J^T(q)F_{ext} \tag{4.11}$$

where $J^T(q)$ is the Jacobian matrix that associates the linear and angular velocity of a frame located at the point where F_{ext} is applied to the joint velocity $\dot{q}$. In this work we assume that external forces only act on the end-effector frame EE_r and EE_l as defined in Figure 4.17. τ_m can me readily measured using the high resolution joint torque sensors on the ballbot arms. However, τ_{ext} cannot be easily calculated by solving equation (4.10), because the joint accelerations $\ddot{q}$ are usually noisy and inaccurate. In our work we implement a residual observer similar to that in [55, 61] to estimate τ_{ext}.

Let the residual r be defined as

$$r(t) = K\left[\int_0^t (C(q,\dot{q})^T\dot{q} - g(q) + \tau_m - r)dt + p\right], \tag{4.12}$$

where the diagonal matrix $K > 0$ is the cutoff frequency of the observer, and $p = M(q)\dot{q}$ is the generalized momentum of the arm. Assuming that the control period is fast enough (> 500 Hz), then $r = \tau_{ext}$. The wrench W acting on the end-effector is obtained by solving equation (4.11)

$$W = J^{T^\dagger}(q)r - W_0 \tag{4.13}$$

where $J^{T^\dagger}(q)$ is the pseudo inverse of $J^T(q)$ and $W_0 = W(t=0)$ is the initial wrench estimate at time $t = 0$. The torque measurements from the joint sensors are noisy. To reduce the propagation of the raw measurement noise to the wrench estimate we implement a simple moving average filter.

4.5 Balancing Controller

The existing balancing controller described in Sec. 2.3.1 can balance the enhanced ballbot when the arms are in their zero configuration. Any movement of the arms' configuration will destabilize the system, and the controller will not maintain balance. We present an extension to the balancing controller to enable arm motion while dynamically balancing. The extension follows the same strategy used in Chapter 3 to lift heavy payloads with the 2-DOF arms. The approach is to control the system's COM position by controlling the desired body lean angle $\boldsymbol{\phi_d} \in \mathbb{R}^2$. This method decouples the arms control from the balancing controller and enables the use of the controllers described in Sec. 4.4 for the arms. Similar to before, the total body lean angle command sent to the balancing controller is:

$$\boldsymbol{\phi}_d = \boldsymbol{\phi}_m + \boldsymbol{\phi}_{fb} + \boldsymbol{\phi}_p. \tag{4.14}$$

With the 2-DOF arms, the ballbot kinematics were projected to the two orthogonal vertical planes, and the COM position was computed for the planar ballbot model. The manipulation body lean angle $\boldsymbol{\phi}_m$ could be easily computed. With the new pair of 7-DOF arms, the robot kinematics cannot be easily expressed in the two orthogonal vertical planes. Instead we first compute the 3D COM position $\vec{p}_G^S = [p_{G,x}^S, p_{G,y}^S, p_{G,z}^S]^T \in \mathbb{R}^3$ with respect to the fixed ball frame $\{S\}$ resulting from the current shape configuration $q_s(t)$. Then the COM position is projected to the frontal and sagittal planes. A schematic of the 3D kinematics and COM projections of the ballbot with a pair of 7-DOF arms is shown in Fig. 4.17. The 3D COM position is computed used the Forward Kinematic map

$$\vec{p}_{G,i} = FK_i(\boldsymbol{q}_s(t)), \quad \forall i = 1...N_{link}, \tag{4.15}$$

that converts robot's shape variables $\boldsymbol{q_s}$ to the COM position $\vec{p}_{G,i}$ of each link i, the ballbot has 15 links in the kinematic structure, *i.e.*, $N_{link} = 15$. The COM of the entire system is computed from

$$\vec{p}_G^S(\boldsymbol{q}_s) = \frac{\sum_i^N m_i \cdot \vec{p}_{G,i}}{\sum_i^N m_i}. \tag{4.16}$$

Note that in this definition, the COM position is only a function of the shape variables (*i.e.*, the body lean angle and the arm joint configuration). It is not a function of the ball's position on the floor because, for balancing, we are only interested in the relative X-Y position of the COM with respect to the point of support that coincides with frame $\{S\}$. The projection of the COM to the vertical planes is necessary because of the decomposition of the balancing controller into these planes. The COM position represented as an angle with respect to the vertical axis is computed as follow:

$$\phi_G^x = \cos^{-1}\left(\frac{p_{G,z}^S}{pG_x{}^S}\right), \tag{4.17}$$

$$\phi_G^y = \cos^{-1}\left(\frac{p_{G,z}^S}{p_{G_y}^S}\right). \tag{4.18}$$

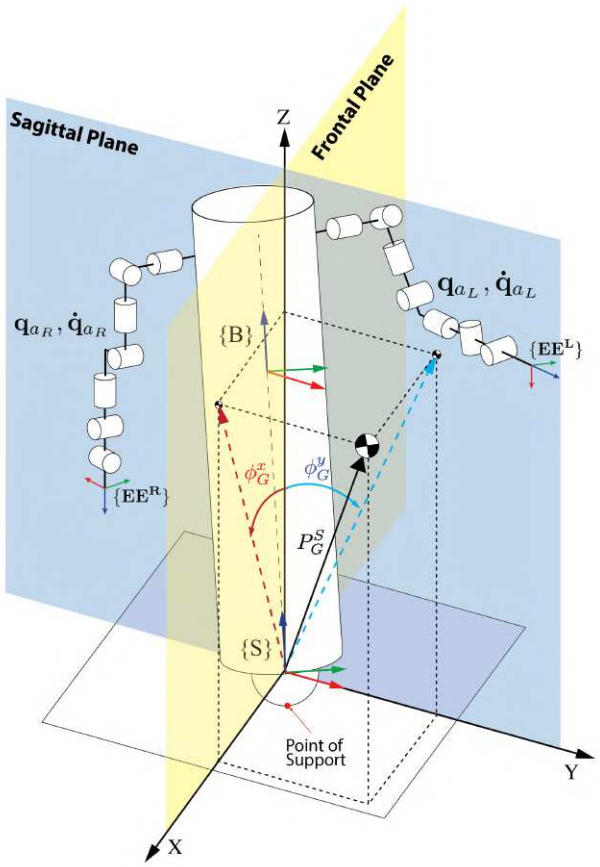


Figure 4.17: Schematic of the ballbot with 7-DOF arms 3D kinematics and projection of the COM to the sagittal and frontal planes.

The body lean angle command $\boldsymbol{\phi}_m = [\phi_m^x, \phi_m^y] \in \mathbb{R}^2$ to compensate for the effects of the upper body motion are computed as

$$\phi_m^x = -\phi_G^x \quad \text{and} \quad \phi_m^y = -\phi_G^y. \tag{4.19}$$

The feedback compensation term $\boldsymbol{\phi}_{fb}$ again is computed from a PD-Control law closing the loop around the ball position. This term is necessary to compensate for the unmodeled dynamics, nonlinear friction effect, and initial condition mismatch on the real robot. With a good model of the ballbot system, this term is usually small. The planned body lean angle $\boldsymbol{\phi}_p$ term is used to realize secondary tasks with the ballbot, such as navigation or exerting a force on the environment.

Tracking the desired body angle was sufficient to minimize the upper body's effects while lifting heavy objects with the 2-DOF arms. In the experiments with the 2-DOF arms, the lifting motion was assumed to be slow, and only significant sudden changes in the COM position happened with the addition/removal of the payload. This assumption is no longer applicable when operating the 7-DOF arms. When swinging the arms around, the COM is constantly changing. The feed-forward term $\boldsymbol{\phi}_m$ alone is no longer sufficient to compensate for the upper body motion.

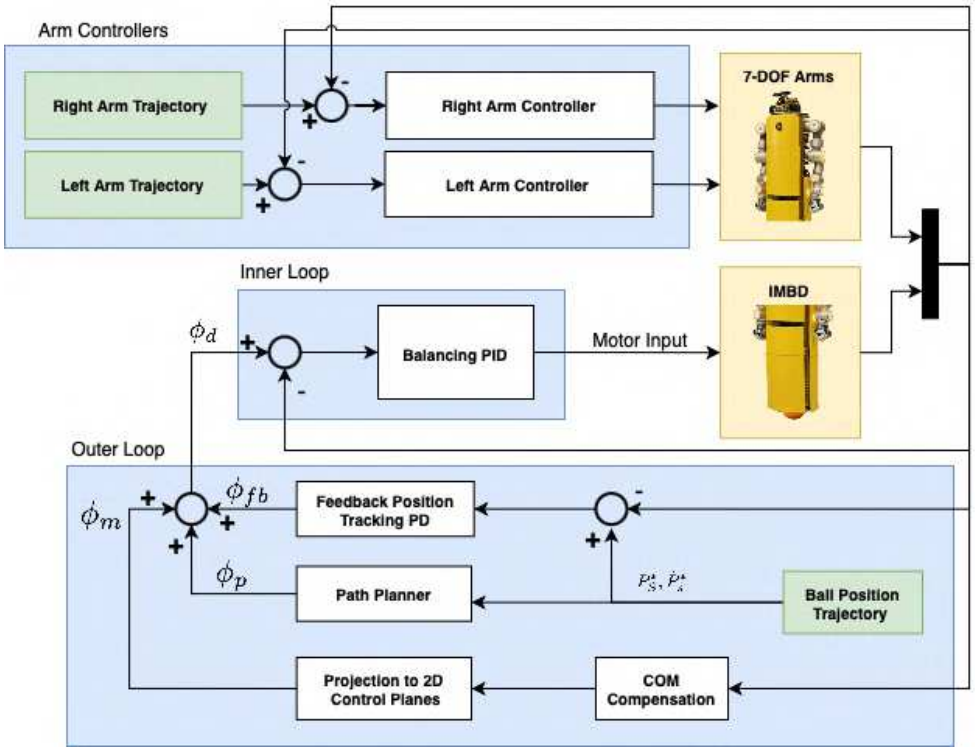


Figure 4.18: Extension of the balancing controller with COM compensation and arm control.

To mitigate this, we introduce a feedback term that outputs a ball velocity command to zero out the body lean angle rate as:

$$\dot{\boldsymbol{\theta}}_d = K_p \dot{\boldsymbol{\phi}}_G. \tag{4.20}$$

The implemented control strategy is shown in Fig. 4.18. Results of the implemented controller are shown in Sec. 4.6.2.

4.6 Experiments and Results

This section presents the results of the experiments conducted to evaluate the new 7-DOF arms and the controllers described in this chapter. The first set of experiments is done with the ballbot in its statically stable configuration. These experiments intend to demonstrate the capabilities of the arms without considering the effects of the ballbot base. The second set of experiments is done with the ballbot dynamically balancing. These experiments show the capabilities of the entire mobile manipulation platform.

4.6.1 Statically Stable Experiments

4.6.1.1 Load Estimation

An important skill for the ballbot is to estimate the mass of the load it is carrying. Mass estimation is possible with the joint torque sensors integrated into the new arms and the wrench estimator described in section 4.4.3. The mass of the payload can be estimated from the equation:

$$m_{load} = F_z/g, \tag{4.21}$$

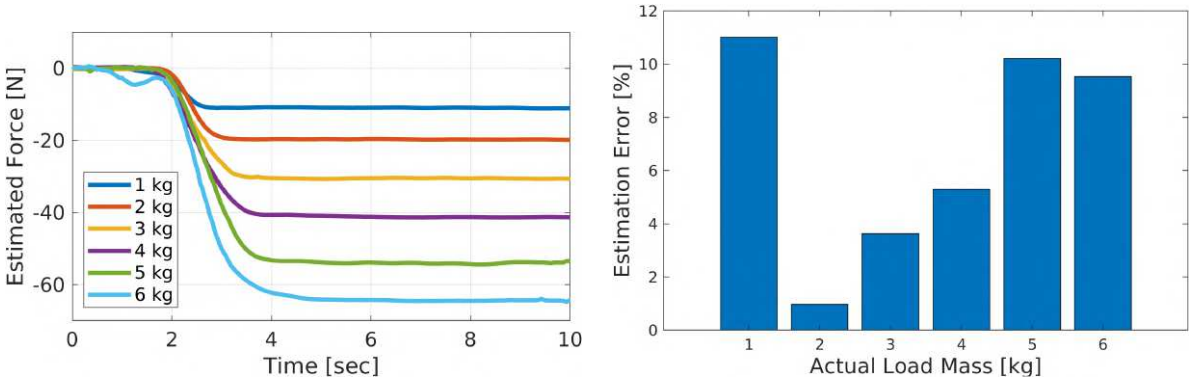
where m_{load} is the estimated payload mass in kg, F_z is the estimated force applied at the end-effector aligned with gravity, and $g = 9.81$ m/s is the acceleration due to gravity. We compared the estimated force to the expected force from lifting a known mass to determine the estimator's accuracy. The experimental setup for this test is shown in Fig. 4.19. The force estimated when

Figure 4.19: Experimental setup to evaluate the wrench estimator accuracy, the ballbot is holding 6 kg with its right arm.

adding different masses to the end-effector are shown in Fig. 4.20 (a). For the six different masses tested, the estimated force was within 11% of the expected force in the worst case and 6.8% on average, as shown in Fig. 4.20 (b). As demonstrated by the experiment in section 4.6.2.4 this mass estimate is sufficient for the ballbot to adapt and balance.

4.6.1.2 Payload Manipulation

The arm's capabilities were evaluated by holding a 6.8 kg dumbbell with the 1.2 kg BH-280 hand while tracking the desired motion. Fig. 4.21 and the Video C.2 show a motion from the zero configuration to elbow flexion, followed by moving the mass above the ballbot's turret and ending with shoulder rotation at full extension. The corresponding joint torque commands are shown in Fig. 4.22. The required torques are significantly lower than the actuator unit limits listed in Table 4.3. Although the arm can lift 10 kg without an end-effector the test was limited to 6.8 kg because it is the maximum payload the BH-280 can hold before its fingers buckle.



(a) (b)

Figure 4.20: Results of the load estimation experiment. (a) shows the estimated force along the direction of gravity for different masses, and (b) shows the error in the estimated force.

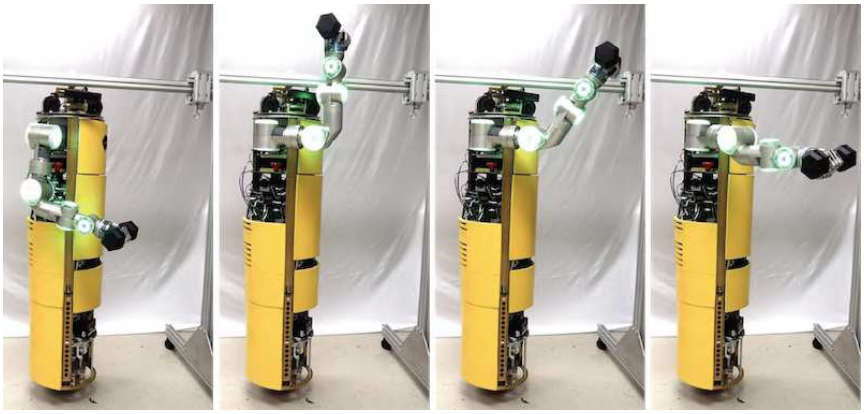


Figure 4.21: Screenshots of the CMU ballbot lifting 6.8 kg dumbbell with the new arm and Barrett Hand. Here, the ballbot is constrained at the top by a fixture and is not balancing.

4.6.1.3 Dual Arm Control

To validate the simultaneous control of the 7-DOF arms integrated into the ballbot, the task of picking a box was considered. The ballot was statically stable in this experiment with its triad of legs deployed. This was done to evaluate the arms' control without the effects of the moving base. To perform this task, the arms were controlled using the joint space impedance controller, described in Sec. 4.4.1, to position the arms close to the box. The task space admittance controller, described in Sec. 4.4.2, takes over to grasp the box and lift it. The controllers were implemented using the ROS-Control framework [38] and ran onboard ballbot on the non-real-time Linux machine. The reference joints position and velocity were sent to the low-level arm position control at 250 Hz. The end-effector trajectory was hand-crafted since the path planning task was outside the scope of this experiment. Fig. 4.23 shows screenshots of the ballbot successfully picking up an empty box using both of its arms from a table in front. The compliance of the arms prevents the box from being squashed by the motion of the arms. The results of this experiment prove that both arms can be controlled simultaneously.

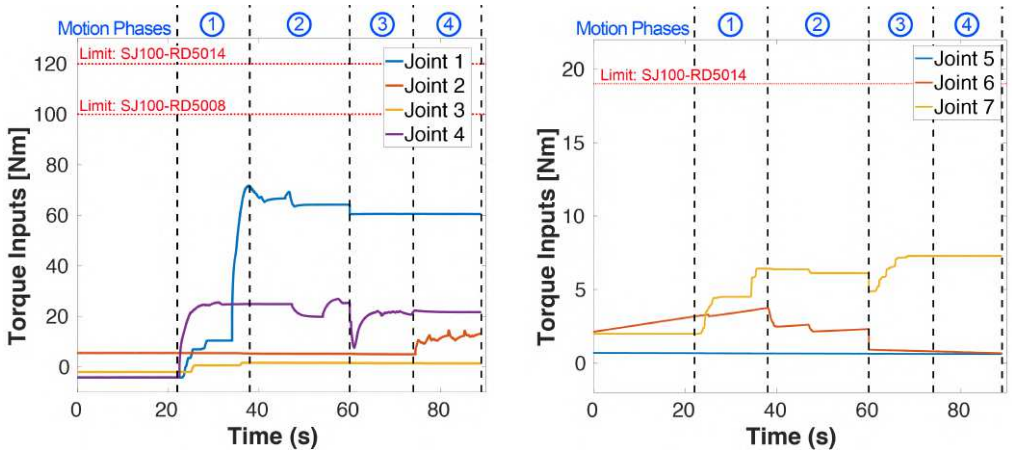


Figure 4.22: Joint torque command evolution to lift a 6.7 kg payload at the end of the arm. The arm is moved from pointing straight down to fully extended, then performs a shoulder rotation at full extension.

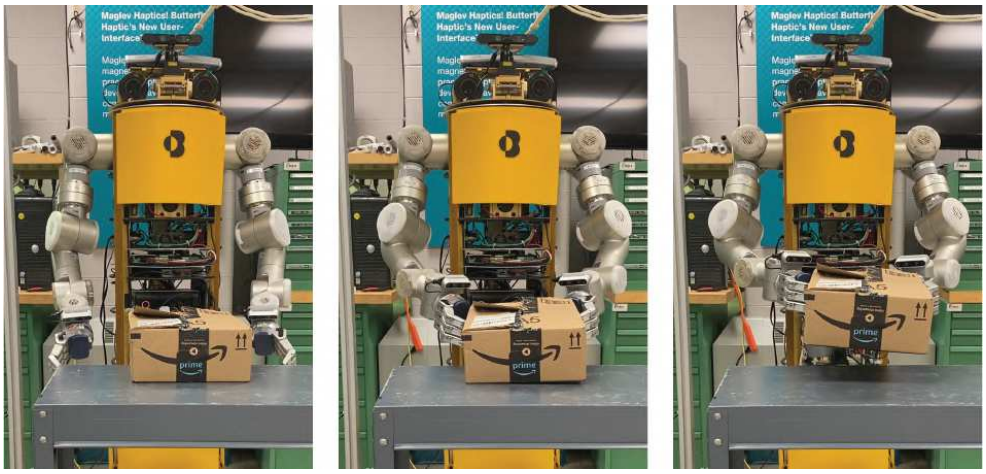


Figure 4.23: Screenshots of the ballbot picking up a box with its 7-DOF arms while statically stable.

4.6.2 Dynamically Stable Experiments

This section presents the results of the proposed balancing controller in Sec. 4.5 implemented on the ballbot with a pair of 7-DOF arms. Balancing with the new 7-DOF arms is a significantly harder challenge than with the previous 2-DOF arms. The new arms cannot be assumed to be massless and will have bigger dynamic effects on the robot's stability. The motion of the 7-DOF arms has a similar effect as carrying a heavy payload with the 2-DOF arms, the COM shifts away from the point of support. Fig. 4.24 shows the controller successfully balancing the ballbot with its 7-DOF arms in a range of different configurations. In addition, to maintain balancing, a fundamental capability is to stay in one place while operating the new 7-DOF arms. The results presented in this section demonstrate this capability. The videos of the ballbot achieving the experimental results presented in this section can be found in Video C.2.

(a) (b) (c)

Figure 4.24: The enhanced CMU ballbot balancing with the arms in different configurations. Here, a slack rope is attached for safety.

4.6.2.1 Single arm motion

In the experiments realized with the 2-DOF arms, both arms moved synchronously and equally. In this experiment with the 7-DOF arms, this constrain is removed. In this experiment, we tested station-keeping while moving the arms asynchronously. One arm was kept stationary at the zero-configuration while the other arm was commanded to full extension as shown in Fig. 4.24 (a). This is a difficult task because the COM shifts the greatest at full arm extension and is not symmetrical about the sagittal plane. The ballbot can maintain its balance throughout the entire motion. With the COM regulator and, the ball position on the floor tracking error decreased by $\sim 87\%$ in the axis of motion, as shown in Fig. 4.25. Also, the oscillation of the ball position is reduced, and the steady-state is reached faster in both axes. The steady-state error with the COM regulation was < 0.15 m for the Y-axis and < 0.05 m for the X-axis. Note the motion of the arm is in the Y-axis direction.

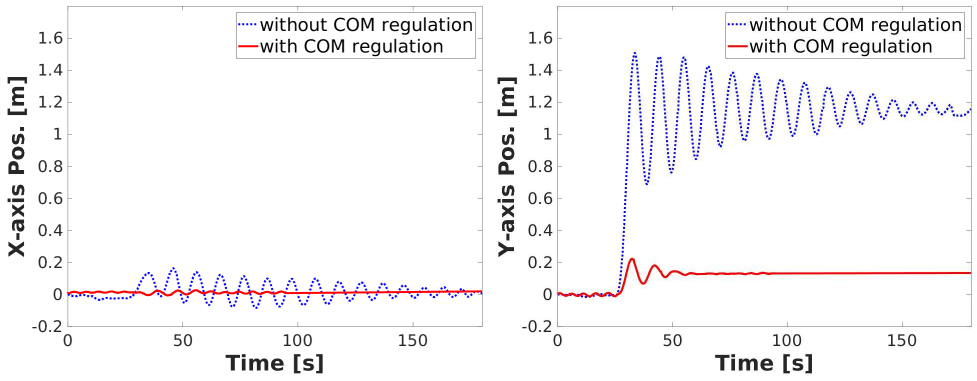


Figure 4.25: The ballbot's X and Y axis position on the floor during the motion of a single arm from straight down to full extension. The outer loop station-keeping controller is enabled in both experiments.

4.6.2.2 Synchronized double arm motion

The motion of both arms synchronously and equally is also a necessary capability for the ballbot. In this experiment, both arms move simultaneously from pointing straight down to a $90°$ elbow flexion while trying to station-keep, as shown in Fig. 4.24 (b). The ballbot can maintain its balance through the entire motion. Fig. 4.26 shows the benefits of the addition of the COM compensation to the balancing controller. With the COM compensation, the ball position tracking error significantly decreases by $\sim 85\%$. Similarly to the single-arm motion, the ball position steady-state error is $<$0.15 m. Further, oscillations and settling time in both axes also decrease. The COM compensation also removes the drift in the lateral (*i.e.*, x-axis) position.

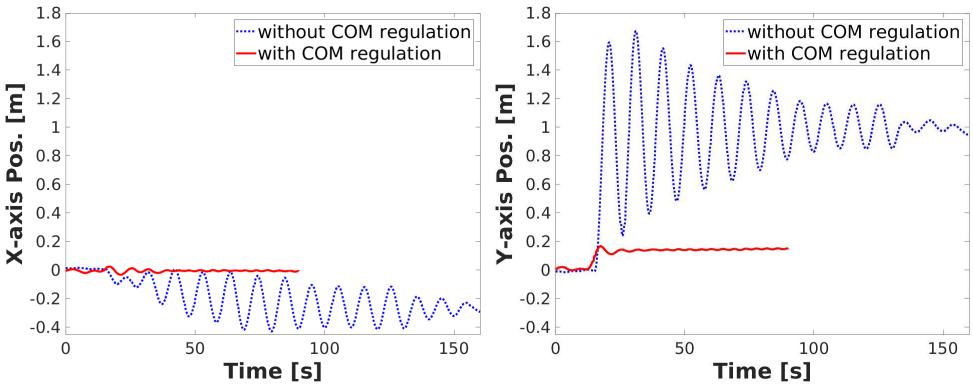


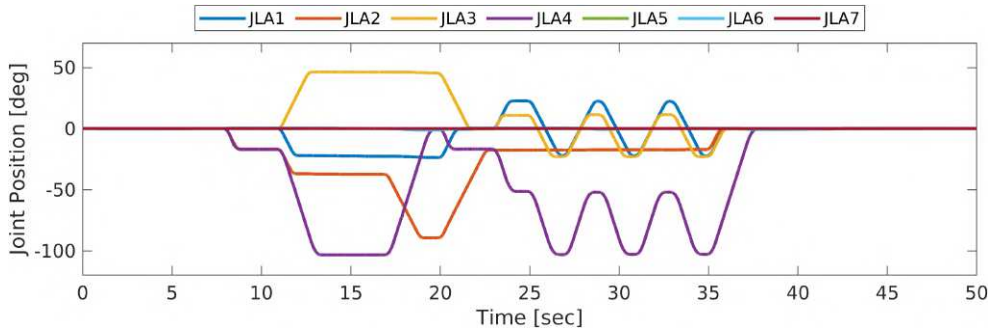
Figure 4.26: The ballbot's X and Y linear position on the floor during the motion of both arms from straight down to $90°$ elbow flexion. The outer loop station-keeping controller is enabled in both experiments.

4.6.2.3 Choreography

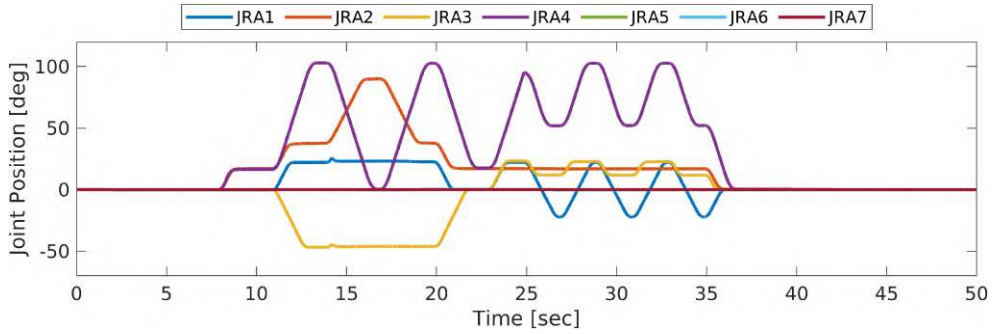
In the previous experiments, the arms motion was simple and relatively slow. In reality, most manipulation tasks will require the actuation of multiple joints simultaneously. The next experiment executes a choreography for the arms while balancing and tracking a fixed floor position to demonstrate this capability. Without the COM compensation strategy, complex arm motions were impossible. The choreography goes through different arm configurations as shown in Fig. 4.27 (a) and (b). The ballbot has to lean up to 2 degrees to compensate for the upper body motion, as shown in Fig. 4.27 (c). As shown in Fig. 4.27 (d), throughout the arm choreography the ballbot position only deviates ± 0.05 m in both axes. This is significantly lower than in the previous experiments with simpler arm motions. This improvement is because new model parameters for the ballbot model were used for this experiment. The updated parameters led to better feedforward body lean angle compensation terms. Screenshots of the entire arm choreography are shown in Fig. 4.28 and the in the Video C.3.

4.6.2.4 Lifting Heavy Payloads

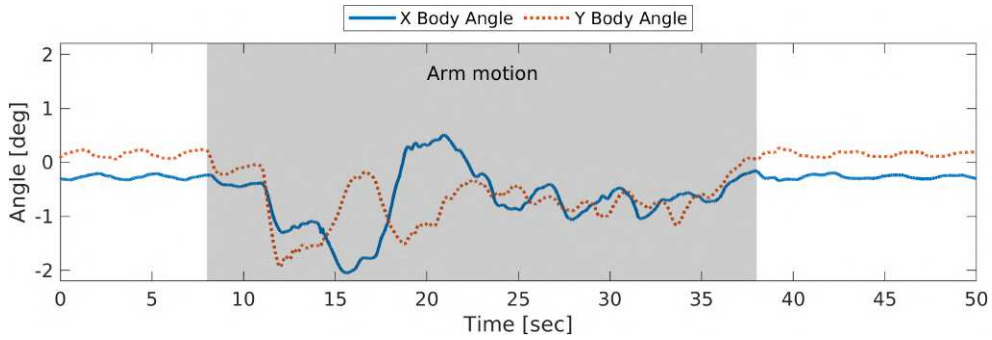
Another important capability for the ballbot is to lift payloads of unknown mass and adapt its posture to maintain balance. This experiment further demonstrates the capabilities of the con-



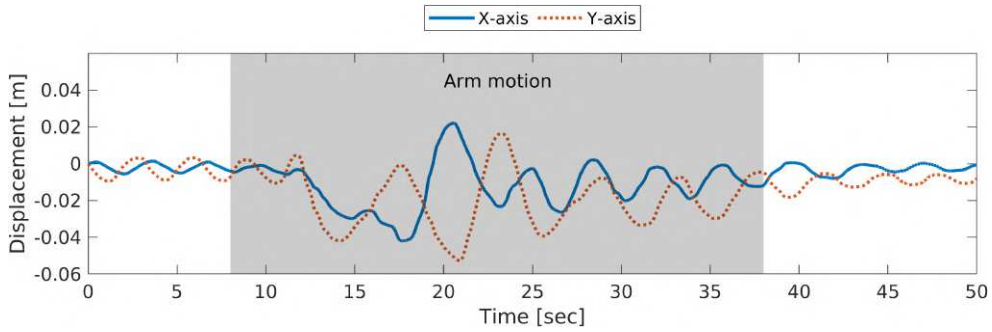
(a)



(b)



(c)



(d)

Figure 4.27: Time series plot of the ballbot's state evolution while station keeping and performing an arm choreography routine. (a) and (b) show the left and right arm state, (c) the body lean angle, and (d) x-y ball displacement.

(a) t = 0 s (b) t = 3 s (c) t = 6 s

(d) t = 9 s (e) t = 12 s (f) t = 18 s

(g) t = 18 s (h) t = 21 s (22) t = 22 s

Figure 4.28: Screenshots of the ballbot station keeping while performing an arm choreography.

troller. The ballbot has to balance while lifting an increasing unknown payload in this task. The controller regulates the ball position on the floor and the arm configuration. Fig. 4.29 (a) shows the experimental setup used for this test. A wood plank is placed across the ballbot's forearms to provide a surface to place the weights. Weights are added until the total payload reaches 15 kg. Without the implemented controller, the ballbot would accelerate forward and becomes unstable. With the controller enabled, the robot maintains its balance and tracks its ball position within ± 0.2 m as shown in Fig. 4.30 (a). The larger ball displacement, in comparison to previous experiments, is expected. Recall the ballbot has no direct control over the body lean angle. The ball must move to induce the body lean angle necessary to compensate for the payload mass. To carry 15 kg the ballbot has to significantly lean (*i.e.*, $4.6°$), as shown in Fig. 4.30 (b). The

(a) (b)

Figure 4.29: Experimental setup for evaluating the ballbot lifting task. In (a) the ballbot is balancing with 15 kg, and in (b) balancing with 21 kg and with a body lean angle of 5.3°.

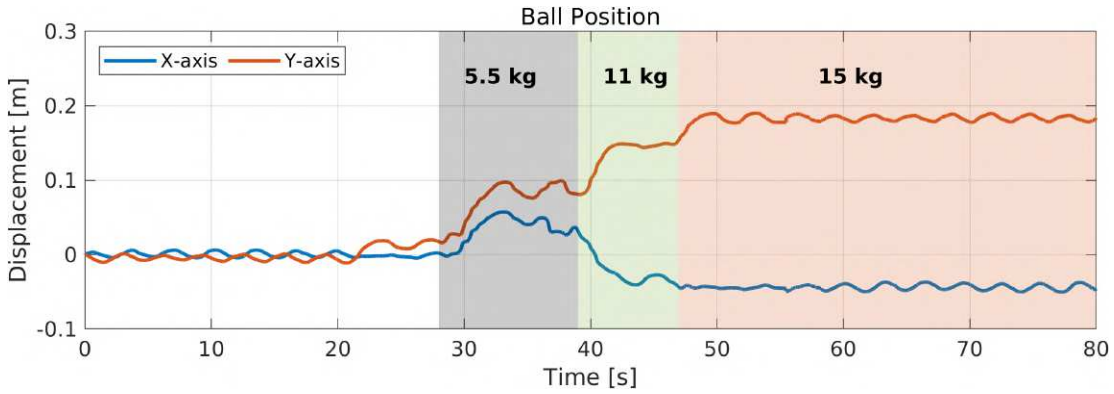
maximum payload that the ballbot was able to carry was 21 kg and required a 5.3° body lean angle, as shown in Fig. 4.29 (b). Experimental results can be seen in the Video C.4.

4.7 Discussion

This chapter introduced a new pair of 7-DOF arms with multi-DOF hands to enhance the CMU ballbot research platform. The developed 7-DOF arms are of comparable size and weight to an average adult human. This chapter also presented the joint and task space controllers implemented in the arms. The controllers were tested by performing simple manipulation tasks while the ballbot was statically stable. These controllers are the foundation for future work to realize complex tasks such as maneuvering a manual wheelchair, leading elderly or sight-impaired individuals from place to place inside a building, and cooperative carrying heavy objects.

The COM compensation controller implemented in this chapter allowed the ballbot with its new 7-DOF arms to balance and station-keep. In the first two experiments the controller had considerable ball position error, *i.e.*, ± 0.15 m. This is not ideal for tasks that require precise end-effector position relative to an inertial frame. However, with a better kinematic model and small gain tuning, the controller performance significantly improved, as demonstrated in the arm choreography experiment. The COM compensation controller only considers the kinematics. Accounting for the dynamic effects of the arm motion can further improve the tracking performance. Additionally, we demonstrated the ballbot capable of adapting to maintain balance while the payload is transferred to it. A maximum carrying payload of 21 kg was attained, this breaks the previous record of carrying a 15 kg payload with the simple 2-DOF arms.

Having demonstrated complete integration of the 7-DOF arms with the ballbot, the following chapter investigates more intelligent planning and controls that combine manipulation and locomotion into a single framework.



(a)

(b)

Figure 4.30: Ballbot state evolution while lifting a varying weight payload. (a) shows the ballbot's ball position on the floor and (b) shows the body lean angle to compensate for the payload mass. Shaded area indicates the total payload weight being lifted (grey: 5.5 kg, green: 11 kg, yellow: 15 kg).

Chapter 5

Task Space Impedance Control

In Chapter 3 the problem of picking up boxes with a dynamically stable shape-accelerating mobile manipulators, like the ballbot with arms, was presented. The control framework was developed and implemented for a simpler ballbot base manipulator system with a pair of 2-DOF instead of 7-DOF arms. Given the lower number of DOF and simpler kinematic tree, it was possible to realize the entire control law in joint space. It was simple to generate pick-up motions in joint space. Further, because the arms were simple lightweight aluminum hollow tubes, their mass and inertia could be considered negligible. Thus, the motion of the arms did not really affect the stability behavior of the ballbot. These assumptions are not valid in the ballbot with a pair of 7-DOF arms.

The tasks space admittance controller described in Sec. 4.4.2 assumed the robot arm was mounted to a fixed base. In reality, the arms are mounted to a mobile base with underactuated DOF. An end-effector (EE) control strategy is necessary to account for the ballbot's ball position and body lean angle. This chapter extends the framework presented in section 4.5 to realize accurate end-effector control with the ballbot with a pair of 7-DOF arms. To control the robot's end-effectors, a floating-base Task Space Impedance Control is introduced that accounts for the motion of the lower body. The control framework uses the same philosophy to decouple the control of the lower body, *i.e.*, body and ball, from the upper body, *i.e.* the pair of arms. The presented framework is tested by performing different tasks with the enhanced CMU ballbot.

5.1 Background

There exist many mobile manipulators both in academia and industry. The goal for these robots is to perform combined manipulation and locomotion tasks in collaboration with humans. To realize this, robots require the skill to accurately control the position of their end-effector when moving in free space and be compliant when it makes contact with the environment. Broadly speaking, end-effector control to enable these types of skills can be grouped into three categories:

Motion Control: The objective is to track a desired end-effector pose or trajectory as accurately as possible. It does not matter what forces are applied by the environment. This is useful when performing repetitive position tasks in free space, such as a pick-and-place task in a constrained factory setting. However, it can be dangerous as there are no bounds on the

force that the robot can apply to the environment and can potentially cause damage.

Force Control: The objective is to track the desired force instead of position. The robot arm will try to maintain the desired force between the end-effector and the environment. It requires a source of force feedback to sense physical interactions, which makes it suitable for human-robot interactive applications. However, it is limited when moving the robot through free space (*i.e.* when there is no applied force).

Compliance Control: The objective is to combine motion and force control to get the benefits of both. There are several different force/position control techniques. We are interested in Impedance control, where a virtual mass-spring-damper system is inserted between desired and actual end-effector positions. This is good when the task requires following the desired pose and reacting compliantly to external disturbances.

5.1.1 Task Space Impedance Control

Impedance control was first introduced by Hogan in the seminal work [62, 63] and is considered as a classical control approach in robotics. The controller's goal is to modulate the mechanical impedance of the manipulator. It does so by manipulating the dynamics of the system through feedback control. Impedance control regulates the relationship between force and position on the one hand and velocity and acceleration on the other hand [64].

Manipulation tasks are usually specified in terms of motion in the *Task space* (Operational space), *i.e.* desired Cartesian coordinates describing the pose or motion of the manipulator's end-effector. If we are interested in enabling dexterous and compliant end-effector motions, the dynamics of the robot must be considered. Joint space dynamic control algorithms only suffice for motion control in the free space [65]. Thus, task space control algorithms become necessary when controlling the interaction between the end-effector and the environment. The Task Space Formulation developed by Khatib [66] provides a framework for controlling *redundant* and *non-redundant* manipulators with respect to the Cartesian coordinates. The framework derives end-effector dynamics for fixed-based rigid-body robot manipulators from the joint space dynamics. Additionally, it implements a nullspace projection technique to treat redundant robots. In typical implementations, the controller has the Cartesian position $\boldsymbol{x} \in \mathbb{R}^6$ and velocity $\dot{\boldsymbol{x}} \in \mathbb{R}^6$ of the end-effector as inputs and gives the motor torques $\boldsymbol{\tau}$ as outputs. This formulation is not sufficient to control the ballbot's end-effectors. The framework has to be extended to consider the floating base, the ballbot mobile base, that the arms are mounted to.

5.1.2 Control of Redundant Robots

A robot is said to be redundant if the number m of task coordinates is smaller than the number n of DOFs of the robot. The inverse kinematic problem will have infinite solutions when a robot is redundant. This implies that, for a given constant pose of the robot's end-effector, it is possible to perform a joint motion without changing the posture of the end-effector. Any joint motion which keeps the end-effector fixed is called a nullspace motion. It is non-trivial to control the remaining $n - m$ redundant DOF to generate nullspace motions.

The problem of controlling fixed-based redundant manipulators has been widely studied in

the robotics literature, and state-of-the-art methods like [66, 67, 68] can define and limit the interaction forces in a reactive manner. However, if the manipulator is mounted on a mobile base, additional control action is required to balance the robotic platform itself. This is the case for underactuated dynamically stable mobile bases, such as humanoids, quadrupeds, or wheeled balancing systems, that require a planning strategy to maintain their balance [4, 5, 6, 69]. The CMU ballbot is an example of such a robot.

Most of the proposed approaches employ an inverse kinematics (IK) method to transform from task space to joint space. In control, it is typical to solve the IK problem instantaneously at the differential (*i.e.* velocity) level, exploiting the linear relationship between the joint-space and task-space velocities using the manipulator's Jacobian matrix. More sophisticated methods can handle inequality constraints by solving the IK problem as a general quadratic program (QP) [70, 71, 72], which may include limits on the joint positions and their derivatives. However, these methods only consider the task of motion control of the end-effector and not the compliant behavior.

Sentis and Khatib [73] developed an extension to the task space impedance control framework for fixed-based manipulators [66] to allow for behavior-oriented whole-body control for legged humanoid robots, based on task prioritization. The framework integrates task-oriented dynamic control and control prioritization [74] to execute multiple task primitives while complying with physical and movement-related constraints. Prioritization establishes a hierarchy between control spaces. It assigns top priority to constraint-handling tasks while projecting operational tasks in the nullspace of the constraints. For a ballbot type manipulator task prioritization is necessary, as balance must be maintained while performing a manipulation task. In our framework, we borrow the idea of projecting lower-priority task Jacobians into the nullspace of higher priority tasks, to embed the manipulation tasks in the nullspace of the balancing task.

Dietrich et al. [75] presented a whole-body impedance controller for the statically stable mobile manipulator Rollin' Justin. The nonholonomy of the wheeled systems prohibits the direct implementation of impedance control due to kinematic rolling constraints that must be taken into account in modeling and control. The framework employs an admittance interface to the kinematically controlled mobile platform and an impedance interface based on potential energy for the torso and arms. The upper body impedance control law, the platform admittance interface, and the compensation of dynamic couplings between both subsystems yield a passive closed-loop system. Our framework uses a similar philosophy to treat the lower and upper body control separately but for a dynamically stable mobile manipulator.

5.1.3 Challenges of Ballbots

The CMU ballbot with its pair of 7-DOF arms is an *underactuated dynamically balancing* system in joint space, but a highly redundant system in the end-effectors' task space. Due to its underactuation, to balance, the ballbot requires a control strategy that coordinates the movement of the arms and body. The problem of controlling each of the CMU ballbot's end-effectors presents several challenges and self-imposed constraints.

The CMU ballbot already has an existing robust balancing controller that renders the base compliant to external disturbances. The controller is based on tracking the desired body lean angle to track the desired COM position indirectly. In section 4.5 this controller was extended

to account for the motion of new 7-DOF arms. We want to leverage this existing controller to realize end-effector task space impedance control. This imposes constraints on the method used for end-effector control. Using the balancing controller with COM compensation limits the choice of command inputs. The balancing controller takes as inputs the desired body lean angle and ball velocity (*i.e.* ϕ_d and $\dot{\boldsymbol{\theta}}_d$). We don't have direct control of the torque commands between the ball and body. Having direct torque control would make the control problem easier. However, this would be at the expense of losing the safety guarantees of the existing balancing controller.

If the arm and base motion could be decoupled, a classical task space control could be implemented for each arm on top of the balancing controller. However, for ballbots, the arm and base motion cannot be decoupled. Consider the planar ballbot with a 2-DOF arm in an equilibrium state dynamically balancing shown in Fig. 5.1 (a). Assume you want to position the end-effector

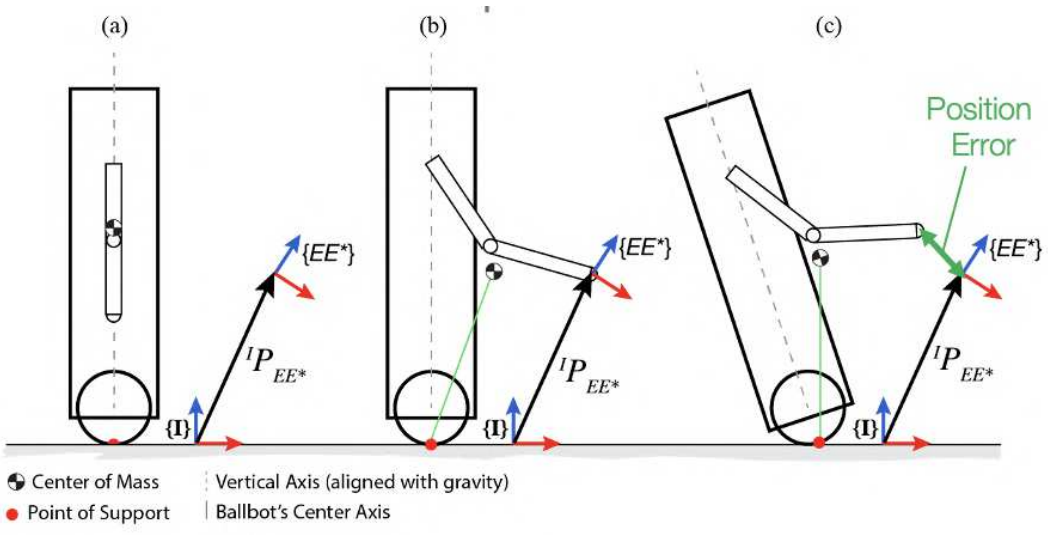


Figure 5.1: Sketch of the the induced end-effector pose error when the ballbot lean angle is not taken into account.

of the arm to be aligned with Frame $\{EE^*\}$. If you move the arm in joint space to reach the desired position it will cause the system's COM to also move forward, as in Fig. 5.1 (b). This will cause the ballbot to accelerate forward and destabilize. The robot has to lean backward to bring the COM back on top of the small patch of support to maintain balance. If the arm does not actively compensate for the body motion, an error will be generated between the desired and actual end-effector. Even when there is no motion of the arms, the ballbot must constantly move its ball to maintain balance. Thus, an end-effector controller has to actively account for the actions of the balancing controller and the movement of its base. Moreover, in the traditional implementation of task space control, all joints in the kinematic chain are controllable. For ballbot's, there is no direct control over the body lean angle DOFs.

5.2 System Model

The joint space dynamics model establishes the manipulator joint motions and provides means for analyzing and controlling these motions. However, the control of the end-effector motion and wrenches requires the development of a model describing the dynamic behavior of this specific part of the robot system [66]. This section introduces the joint space dynamics and the corresponding transformation into the task space (operational space) dynamics. The joint forces, torques, velocity, and acceleration are replaced with the end-effector wrench, twist, and time derivative. For the derivation of the controller in section 5.3 the model makes the following assumptions:

1. There is no slip between the ball and the floor.
2. The ball height relative to the floor remains constant, *i.e.*, the ball is always in contact with the floor.
3. There is now yaw motion between the floor and the ball.
4. There is no reactive friction forces acting on the ball position $\boldsymbol{P}_s$ and body lean angle $\boldsymbol{\phi}$ DOFs.

5.2.1 Kinematic Model

The control framework presented uses a free-floating model of the CMU ballbot, where five virtual unactuated DOF describe the dynamics of the free-floating base (*i.e.* the IMBD link), as shown in Fig. 5.2. Usually, free-floating bases models have six virtual unactuted DOF, but we drop the z-axis (vertical) position since we assume that the ball is always in contact with the floor. Thus, the robot's task and posture kinematics are defined with respect to the origin of the free-floating model (*i.e.* the inertial frame $\{I\}$). The relationship between the task coordinates $\boldsymbol{x} \in \mathbb{R}^m$ and the joint configuration coordinates $\boldsymbol{q} \in \mathbb{R}^n$ is given by the forward kinematics function $\mathbf{FK} : \mathbb{R}^n \rightarrow \mathbb{R}^m$,

$$\boldsymbol{x} = \mathbf{FK}(\boldsymbol{q}_b, \boldsymbol{q}_a), \tag{5.1}$$

where $\boldsymbol{q} = [\boldsymbol{q}_b, \boldsymbol{q}_a]^T$, $\boldsymbol{q}_b \in \mathbb{R}^5$ is a vector of the floating base joints, and $\boldsymbol{q}_a \in \mathbb{R}^{14}$ is the vector of the actuated arm joints. This kinematic description allows accounting for the body lean angle and position effect on the end-effector position. Classical task space control methods define the kinematics with respect to the base frame of the manipulator, where all joints in the tree are actuated. The formulation presented is with respect to inertial frame $\{I\}$ and has passive joints in the kinematic tree.

5.2.2 Joint Space Dynamics

The configuration of the robot is described by $n = n_b + n_a$ DOFs, where n_b is the number of DOF of the mobile base and n_a is the DOF of the manipulators attached. The motion of the ballbot base can be described by $\mathbf{q}_b = [\mathbf{P}_S, \boldsymbol{\phi}]^T \in \mathbb{R}^{n_b}$ where $n_b = 5$. $\mathbf{P}_S = [p_x, p_y]^T \in \mathbb{R}^2$ is the 2D position of the ball center in the horizontal plane with respect to the inertial frame $\{I\}$, $\boldsymbol{\phi} = [\phi_x, \phi_y, \phi_z]^T \in \mathbb{R}^3$ is a vector of Euler angles representing the lean angle of the body with

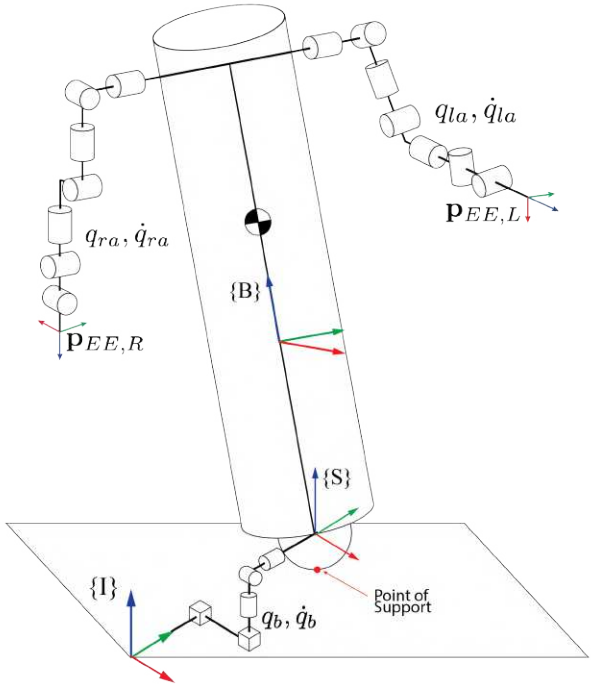


Figure 5.2: Schematic of the CMU ballbot floating-base kinematic model representation. $\{I\}, \{S\}, \{B\}$ are the inertial frame, the ball (sphere) frame and the body frame. $\{EE^R\}$ and $\{EE^L\}$ are right and left end-effector frames.Cubes represent linear joints and cylinders revolute joints.

respect to the gravity vector and the body yaw around the vertical axis. The body lean and yaw angles are measured directly from the IMU. The ball position (p_x, p_y) are function of the ball angles (θ_x, θ_y) such that

$$\begin{aligned} p_x &= r(\theta_x + \phi_x), \\ p_y &= r(\theta_y + \phi_x), \end{aligned} \tag{5.2}$$

where r is the radius of the ball. The position The 7-DOF of the arm are represented by $\mathbf{q}_a \in \mathbb{R}^{n_a}$. For the CMU ballbot with the set of generalized coordinates

$$\mathbf{q} = \begin{bmatrix} \mathbf{P}_S \\ \boldsymbol{\phi} \\ \mathbf{q}_a \end{bmatrix} \text{ and } \dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{P}}_S \\ \dot{\boldsymbol{\phi}} \\ \dot{\mathbf{q}}_a \end{bmatrix} \tag{5.3}$$

is defined. The system has in total $n = 19$ DOF (*i.e.* $n_b = 5, n_a = 14$). Recall, the ballbot is *underactuated* and the body lean angle DOFs (ϕ_x, ϕ_y) are passive, *i.e.* it has $n_c = 17$ active joints and $n_p = n - n_c = 2$ passive joints.

The rigid-body dynamics equation can be written as:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\epsilon} = \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}^T(q)\boldsymbol{F}_{ext} \tag{5.4}$$

where $\boldsymbol{M}(q) \in \mathbb{R}^{n \times n}$ is the mass/inertia matrix, $\boldsymbol{h}(q, \dot{q}) \in \mathbb{R}^n$ is the vector of Coriolis, centripetal, and gravity forces, $\boldsymbol{\epsilon} \in \mathbb{R}^n$ is the vector that collects unmodeled arm dynamics (*e.g.*

joint friction forces), $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of joint torques, $\boldsymbol{F}_{ext} \in \mathbb{R}^6$ is the vector of external forces and moments acting at the end-effector, and $\boldsymbol{S} \in \mathbb{R}^{n \times n_c}$ is the selection matrix that separates the n_c controlled joints from the n_p passive body lean angle joints.

5.2.3 Task Space Dynamics

The end-effector motion results from forces and moments acting along or about the axes of displacement or rotation. We define the task in terms of the robot's end-effector motion and wrench for this framework. In general, the configuration of a single end-effector is represented by a homogeneous transformation (*i.e.* as an element of the group $SE(3)$). For the purpose of the controller design, a minimal vector representation $\boldsymbol{x} \in \mathbb{R}^m$ is chosen, where m corresponds to the number of end-effector coordinates of interest. In case all DOFs of a single end-effector motion are considered $m = 6$. Thus, the robot is underactuated in joint space, but redundant in task space (*i.e.* $m \leq n_c$). The derivation of the end-effector dynamic model is achieved by expressing the relationship between its position, velocities, acceleration, and virtual forces acting on it. The relationship between the Cartesian coordinates $\boldsymbol{x}$ and the joint configuration coordinates $\boldsymbol{q}$ is given by (5.1). The differential kinematics that relates the joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ to the task velocity $\dot{\boldsymbol{x}} \in \mathbb{R}^m$ is given by:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{5.5}$$

where the task Jacobian is defined as:

$$\boldsymbol{J}(q) = \frac{\delta \mathbf{FK}(q)}{\delta \boldsymbol{q}} \in \mathbb{R}^{n \times m}. \tag{5.6}$$

Taking the derivative of (5.5) the relation between joint acceleration $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ to the task acceleration $\ddot{\boldsymbol{x}} \in \mathbb{R}^m$ is obtained as:

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{5.7}$$

The Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$ also relates the external torque vector $\boldsymbol{\tau}_{ext}$ to the generalized external forces $\boldsymbol{F}_{ext}$ by

$$\boldsymbol{\tau}_{ext} = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{F}_{ext}. \tag{5.8}$$

Following a similar process as in [64, 66] the joint space dynamics (5.4) are derived in task space as:

$$\boldsymbol{\Lambda}\ddot{\boldsymbol{x}} + \boldsymbol{\mu} = \boldsymbol{F}_{\tau} + \boldsymbol{F}_{ext}, \tag{5.9}$$

where $\boldsymbol{\Lambda} = (\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T)^{-1}$, $\boldsymbol{\mu} = \boldsymbol{\Lambda}(\boldsymbol{J}\boldsymbol{M}^{-1}(\boldsymbol{h} - \boldsymbol{\tau}_{fric}) - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}})$, $\boldsymbol{F}_{\tau}$ is a virtual force at the end-effector related to the joint torques by $\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{F}_{\tau}$, and $\boldsymbol{F}_{ext}$ is the external force applied at the end-effector. For a complete derivation of the task space dynamics refer to Appendix B.

5.3 Control

As originally presented by [66] the relation between task and joint forces consistent with the end-effector and robot dynamic equation is given by:

$$\boldsymbol{\tau}^* = \boldsymbol{J}^T\boldsymbol{F}^* \tag{5.10}$$

where $\boldsymbol{F}^* = \boldsymbol{F}_\tau$ is defined from (5.9) using a desired task acceleration $\ddot{\boldsymbol{x}}^*$ instead of $\ddot{\boldsymbol{x}}$. Equation (5.10) provides a method for task-based force control. As such, it projects task space forces $\boldsymbol{F}^*$ into joint torques $\boldsymbol{\tau}^*$ through the task forward kinematics $\boldsymbol{J}$. This relationship forms the basis for the actual control of the robot in task space. Then the objective becomes to determine the value of $\boldsymbol{F}^*$ to realize the desired end-effector behaviors (*i.e. motion control, force control*, or *compliance control*). This section presents the derivation of different formulations of $\boldsymbol{F}^*$ to achieve the different desired behaviors.

5.3.1 Motion Control

In Motion Control, the objective is to control the ballbot arm's end-effector Cartesian pose. A regulating PD law is used to obtain the desired task acceleration $\ddot{\boldsymbol{x}}^*$ from the current and desired end-effector pose and velocity. The Cartesian pose errors are composed of linear and angular components:

$$\boldsymbol{e}_x = \begin{bmatrix} e_l \\ e_\theta \end{bmatrix} \begin{matrix} \in \mathbb{R}^3, \text{linear error} \\ \in \mathbb{R}^3, \text{angular error} \end{matrix}. \tag{5.11}$$

The linear errors e_l are computed using Euclidean distance for each translation axis as:

$$e_l = p - p_d, \tag{5.12}$$

where $p \in \mathbb{R}^3$ and $p_d \in \mathbb{R}^3$ are the actual and desired position vector of the end-effector with respect to frame $\{I\}$. The angular component error is defined as

$$e_\theta = R_{ee}\epsilon_d^e \in \mathbb{R}^{3\times 1} \tag{5.13}$$

where $R_{ee} \in \mathbb{R}^{3\times 3}$ is the rotation matrix for the end-effector with respect to the inertial frame $\{I\}$, and ϵ_d^e is the vector part of the unit quaternion extracted from R_d^e [60] defined as,

$$R_d^e = R_{ee}^\top R_d, \tag{5.14}$$

the error between the desired end-effector rotation $R_d \in \mathbb{R}^{3\times 3}$ and the actual rotation R_{ee}. The error derivative is defined as:

$$\dot{\boldsymbol{e}}_x = \frac{d\boldsymbol{e}_x}{dt}, \tag{5.15}$$

and in practice we compute it numerically. The desired Task acceleration is obtain as:

$$\ddot{x}^* = \bar{\boldsymbol{K}}_d\boldsymbol{e}_x + \bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x, \tag{5.16}$$

where $\bar{\boldsymbol{K}}_d$ and $\bar{\boldsymbol{B}}_d$ are diagonal proportional and derivative gain matrices. Plugging the desired acceleration (5.16) into equation (5.9) we get

$$\boldsymbol{\Lambda}\left(\bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x + \bar{\boldsymbol{K}}_d\boldsymbol{e}_x\right) + \boldsymbol{\mu} = \boldsymbol{F}_\tau^* + \boldsymbol{F}_{ext}, \tag{5.17}$$

and solving for the desired virtual force to get

$$\boldsymbol{F}_\tau^* = \boldsymbol{\Lambda}\left(\bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x + \bar{\boldsymbol{K}}_d\boldsymbol{e}_x\right) + \boldsymbol{\mu} - \boldsymbol{F}_{ext}. \tag{5.18}$$

Substituting back into (5.10) the final control law in joint torque space is

$$\boldsymbol{\tau}^* = \boldsymbol{J}^T\left[\boldsymbol{\Lambda}\left(\bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x + \bar{\boldsymbol{K}}_d\boldsymbol{e}_x\right) + \boldsymbol{\mu} - \boldsymbol{F}_{ext}\right]. \tag{5.19}$$

5.3.2 Force Control

In Force Control the objective is to control the ballbot arm's end-effector applied force. The force error and error derivative are defined as:

$$\boldsymbol{e}_f = F_{ext} - F_d, \tag{5.20}$$

$$\dot{\boldsymbol{e}}_f = \dot{F}_{ext}, \tag{5.21}$$

where F_d is the desired wrench to be applied. Similar to in motion control, a PD law is used to determine the desired task acceleration as:

$$\ddot{x}^* = \bar{\boldsymbol{K}}_{f,p}\boldsymbol{e}_f + \bar{\boldsymbol{K}}_{f,d}\dot{\boldsymbol{e}}_f, \tag{5.22}$$

where $\bar{\boldsymbol{K}}_{f,p}$ and $\bar{\boldsymbol{K}}_{f,d}$ are diagonal proportional and derivative gain matrices. Plugging back in (5.9) to get

$$\boldsymbol{\Lambda}\left(\bar{\boldsymbol{K}}_{f,p}\boldsymbol{e}_f + \bar{\boldsymbol{K}}_{f,d}\dot{\boldsymbol{e}}_f\right) + \boldsymbol{\mu} = \boldsymbol{F}_\tau^* + \boldsymbol{F}_{ext}, \tag{5.23}$$

and solving for the desired virtual force to get

$$\boldsymbol{F}_\tau^* = \boldsymbol{\Lambda}\left(\bar{\boldsymbol{K}}_{f,p}\boldsymbol{e}_f + \bar{\boldsymbol{K}}_{f,d}\dot{\boldsymbol{e}}_f\right) + \boldsymbol{\mu} - \boldsymbol{F}_{ext}. \tag{5.24}$$

Substituting back into (5.10) the final control law in joint torque space is

$$\boldsymbol{\tau}^* = \boldsymbol{J}^T\left[\boldsymbol{\Lambda}\left(\bar{\boldsymbol{K}}_{f,p}\boldsymbol{e}_f + \bar{\boldsymbol{K}}_{f,d}\dot{\boldsymbol{e}}_f\right) + \boldsymbol{\mu} - \boldsymbol{F}_{ext}\right]. \tag{5.25}$$

5.3.3 Compliance Control

In Compliance Control the objective is to control the ballbot arm's impedance in the Cartesian space of the end-effector to provide a stable physical interaction. In order to specify the desired impedance behaviour the end-effector pose error and its time derivatives are defined as in (5.11) and (5.15). Similarly, the force applied error at the end-effector is defined as (5.20) and (5.21). Then the control objective is to render the dynamic relationship between the external force F_{ext} and the end-effector position error e_x to mimic that of a *mass-spring-damper system* of the form:

$$\bar{\boldsymbol{M}}_d\ddot{\boldsymbol{e}}_x + \bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x + \bar{\boldsymbol{K}}_d\boldsymbol{e}_x = \boldsymbol{e}_f. \tag{5.26}$$

The symmetric positive definite matrices $\bar{K}_d$, $\bar{B}_d$, and $\bar{M}_d$ are the desired stiffness, damping, and inertia matrix, respectively. From (5.26) the task acceleration is:

$$\ddot{x}^* = \ddot{x}_d + \bar{\boldsymbol{M}}_d^{-1}(\boldsymbol{e}_f - \bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x - \bar{\boldsymbol{K}}_d\boldsymbol{e}_x). \tag{5.27}$$

Plugging the desired acceleration (5.27) into equation (5.9) we get

$$\boldsymbol{\Lambda}\left(\ddot{\boldsymbol{x}}_d + \bar{\boldsymbol{M}}_d^{-1}(\boldsymbol{e}_f - \bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x - \bar{\boldsymbol{K}}_d\boldsymbol{e}_x)\right) + \boldsymbol{\mu} = \boldsymbol{F}_\tau^* + \boldsymbol{F}_{ext}, \tag{5.28}$$

and solving for the desired virtual force to get

$$\boldsymbol{F}_\tau^* = \boldsymbol{\Lambda}\ddot{\boldsymbol{x}}_d - \boldsymbol{\Lambda}\bar{\boldsymbol{M}}_d^{-1}(\bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x + \bar{\boldsymbol{K}}_d\boldsymbol{e}_x + \boldsymbol{e}_f) + \boldsymbol{\mu} - \boldsymbol{F}_{ext}. \tag{5.29}$$

Substituting back into (5.10) the final control law in joint torque space is

$$\boldsymbol{\tau}^* = \boldsymbol{J}^T\left[\boldsymbol{\Lambda}\ddot{\boldsymbol{x}}_d - \boldsymbol{\Lambda}\bar{\boldsymbol{M}}_d^{-1}(\bar{\boldsymbol{B}}_d\dot{\boldsymbol{e}}_x + \bar{\boldsymbol{K}}_d\boldsymbol{e}_x + \boldsymbol{e}_f) + \boldsymbol{\mu} - \boldsymbol{F}_{ext}\right]. \tag{5.30}$$

5.3.4 Nullspace Compliance

One ballbot's arm's configuration cannot be specified solely by the end-effector position and orientation coordinates. The solution to the control law (5.10) is not unique. The robot configuration has $n - m$ redundant DOF and can be used for *nullspace motion* that do not affect the motion of the EE. To control these redundant DOF the control law (5.10) can be augmented with the nullspace projection [66] to:

$$\boldsymbol{\tau} = \boldsymbol{J}^T \boldsymbol{F} + (\boldsymbol{I} - \boldsymbol{J}^T \boldsymbol{J}^{T\#}) \boldsymbol{\tau}_{null} \tag{5.31}$$

where $J^{T\#}$ is the generalized pseudo-inverse of J^T:

$$\boldsymbol{J}^{T\#} = (\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T)^{-1}\boldsymbol{J}\boldsymbol{M}^{-1}. \tag{5.32}$$

This generalized pseudo-inverse is dynamically consistent with the task (*i.e.* it is the generalized inverse that results in zero end-effector acceleration). The incorporation of the nullspace projection in (5.31) allows realizing secondary tasks through the joint torque inputs τ_{null} without affecting the primary tasks to track the desired Task Space dynamics.

We choose a value for τ_{null} that regulates the arm joints to a desired position $q_{a,d}$. We define the nullspace torque as:

$$\boldsymbol{\tau}_{null} = \boldsymbol{K}_{q_a,p}(q_a - q_{a,d}) - \boldsymbol{K}_{q_a,d}\dot{q}_a \tag{5.33}$$

where $\boldsymbol{K}_{q_a,p} \in \mathbb{R}^{n\times n}$ is a positive definite diagonal proportional gain matrix, and $\boldsymbol{K}_{q_a,d} \in \mathbb{R}^{n\times n}$ is a positive definite diagonal derivative gain matrix. Incorporating this joint regulation term helps avoid the arm moving towards a singularity configuration.

5.3.5 Integration to Balancing Controller

In section 4.5 we presented a control framework that enabled the ballbot to arbitrarily move a single or both arms in joint space while tracking the desired ball position and maintaining balance. The framework decoupled the control of the lower body, *i.e.*, body and ball, separately from the upper body, *i.e.* the pair of arms controllers. This approach introduces pre-defined levels of hierarchies to the control framework. The lower body controller reacts to the motion of the arms, but the upper body controller does not react to the motion of the lower body. This was a valid approach since the objective was to track the desired ball position and not an end-effector position. The framework did not care about the end-effector's motion relative to an inertial frame. It cared about the internal joint configuration of the robot.

Here, the joint space control law of the arms is replaced with two instances of the task space control law (5.31), one for each arm. Depending on desired task to perform (5.19), (5.25), or (5.30) is used. Fig. 5.3 shows the block diagram of the combined lower and upper body controller. The aim is that both controllers will minimize their respective tracking errors in a close-loop. The lower body balancing controller will continue to react to the disturbances from the movement of the arms. The inclusion of whole-body kinematics to the task space formulation enables the upper body to respond to the body movements. Now, both ball and end-effector targets can be tracked. In case the task is to station keep the ball (*i.e.* track a fixed ball position), then the body lean angle commands are generated similarly as in section 4.5. If, instead, the task is to track a ball position trajectory, a body lean angle trajectory command must be planned.

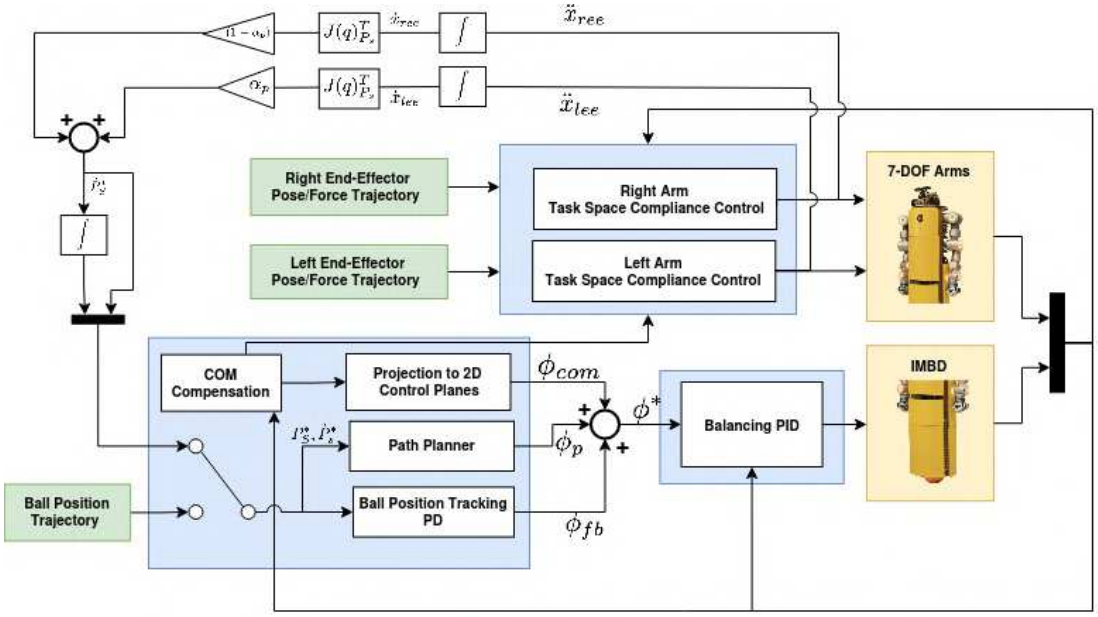


Figure 5.3: Block diagram of the implemented Task Space Impedance Controller on the ballbot with a pair of 7-DOF arms.

5.3.6 Ball Position Controller

The control laws presented in sections 5.3.1 to 5.3.3 output torque commands for all active joints in the floating base model described in section 5.2. This includes the torque commands to actuate the ball. As discussed earlier, we do not have direct control over the torque commands applied to the ball. The COM compensation controller drives the robot's COM to an equilibrium position (*i.e.* to be on top of the point of support where ball position has zero acceleration and velocity) by controlling the body lean angle. Adding a lean angle deviation target will cause the robot to accelerate. As shown in Fig. 5.3 we define the desired body lean angle to track by the lower level balancing controller to be:

$$\boldsymbol{\phi}^* = \boldsymbol{\phi}_{com} + \boldsymbol{\phi}_{fb} + \boldsymbol{\phi}_p, \tag{5.34}$$

where $\boldsymbol{\phi}_{com} \in \mathbb{R}^2$ is the body lean angles output from the COM compensation controller, and $\boldsymbol{\phi}_{fb} \in \mathbb{R}^2$ is the body lean angles generated by a feedback controller on the ball position. The term $\boldsymbol{\phi}_p$ is a body lean angle trajectory planned to track the desired ball trajectory $\boldsymbol{P}_s^*$. To control the movement of the ball position, we implemented two methods.

5.3.6.1 Method 1: External Commands

In this method, the ball position trajectory is generated by some external planner that is decoupled from the arms motion. The ball position trajectory is as a function of time as:

$$\boldsymbol{P}_S^* = f(t), t \in [0, T] \tag{5.35}$$

A path planner is required to transform the desired ball trajectory to a body lean angle trajectory. The differential flatness-based planner presented by Shomin in [76] can be used to generate the

body lean angle trajectory. If the task is instead to stay in one place (*i.e.*, station keep), we can eliminate the use of a path planner. The feedback term $\boldsymbol{\phi}_{fb}$ is sufficient.

5.3.6.2 Method 2: Integration Commands

The previous method is limited because it does not account for the desired end-effector pose. Assume the scenario where the end-effector target pose is outside the reach of the arm. The ballbot has to navigate to reach the target. Integrating the desired task space acceleration $\ddot{\boldsymbol{x}}^*$ from (5.16),(5.22), and (5.27) we get desired task velocity $\dot{\boldsymbol{x}}^*$ commands. Using the differential kinematics formulation we obtain the corresponding ball velocity commands.

$$\dot{\boldsymbol{P}}_S = \boldsymbol{J}(q)_{P_s}^T \dot{\boldsymbol{x}}^*, \tag{5.36}$$

where $\boldsymbol{J}(q)_{P_s}$ is the jacobian matrix that relates the end-effector pose to the ball position coordinates. Ball position commands $\boldsymbol{P}_s^*$ are obtained by integration $\dot{\boldsymbol{P}}_S$ using Euler integration. The ball position and velocity commands are sent to the ball trajectory tracking controller that generates body lean angle $\boldsymbol{\phi}_{fb}$ commands, as shown in Fig. 5.3. Note that if the arms have conflicting end-effector poses that drive the robot in the opposite direction it can destabilize the system. Thus, we introduce the gain variable $\alpha_P \in [0, 1]$ to weight each end-effector controller implementation command.

5.3.7 Minimum Jerk Trajectories

The proposed task space controller for manipulation tasks takes the desired Cartesian pose targets as input. Spline interpolation is used for smooth motions when a big Cartesian target change is specified. A minimum jerk trajectory was used for position interpolation with terminal null velocities and accelerations. The reference position spline $p_r(t), t \in [0, T]$ is a 6^{th} order (*i.e.* C5) differentiable spline:

$$p_r(t) = p_i + (p_f - p_i)\left(10\left(\frac{t}{d}\right)^3 - 15\left(\frac{t}{d}\right)^4 + 6\left(\frac{t}{d}\right)^5\right), \tag{5.37}$$

where p_i is the current position, p_f is the final desired position, t is current query time, d is the desired duration of the motion. This can be set to a predefined value or can be computed using an average desired velocity for the entire motion. The velocity trajectory is obtained by time differentiating (5.37):

$$\dot{p}_r(t) = \frac{1}{d}(p_f - p_i)\left(30\left(\frac{t}{d}\right)^2 - 60\left(\frac{t}{d}\right)^3 + 30\left(\frac{t}{d}\right)^4\right). \tag{5.38}$$

For orientation we employ spherical linear interpolation (slerp) to generate a trajectory between the initial and final desired orientation represented as quaternions.

5.4 Experiments and Results

This section presents results that characterize the performance of the proposed task space impedance control framework on the CMU ballbot hardware. The videos of the ballbot showing the experimental results can be found in the Video C.5.

5.4.1 Experimental Setup

All the experiments were performed on the CMU ballbot with its pair of 7-DOF arms. The controllers were implemented in C++ wrapped around a ROS interface. The task space controller runs at 250 Hz on the Intel Core i7 2.6 GHz CPU running Ubunutu 16.04 and sends commands to each joint motor driver board in the arms. An external sensor is used to measure the pose tracking accuracy of the controller. A Fiducial marker was attached to the end-effector palm, and the pose of the marker was tracked using the open-source ArUco library [77]. An Intel RealSense D435 camera [1] was used as the image sensor. The experimental setup is shown in Fig. 5.4.

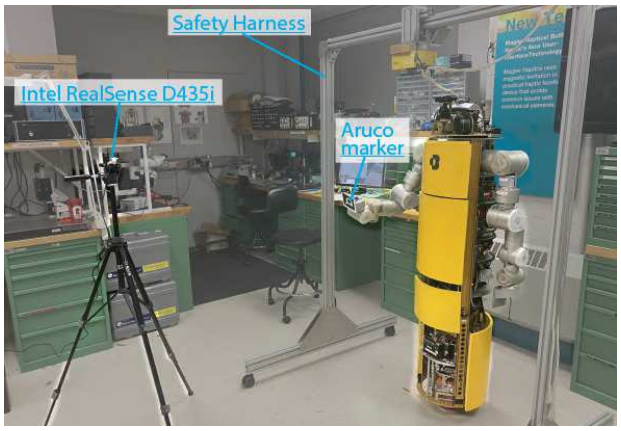


Figure 5.4: Experiment setup to test the motion tracking performance of the task space controller implemented on the CMU Ballbot.

5.4.2 Motion Control

This set of experiments evaluates the position tracking performance of the task space control. In the first experiment, the task is to smoothly move the robot's right end-effector from an initial configuration to a target configuration. The second experiment regulates the arms postures to track a desired end-effector pose when the robot is disturbed. For this set of experiments, we use the motion control law (5.19).

5.4.2.1 Experiment 1: End-effector Trajectory Tracking

In this first experiment, only the right arm is active, and the left arm joints are locked. The task is to track a continuous minimum jerk trajectory between an initial pose $\mathbf{P}_0$ and the desired

[1] https://www.intelrealsense.com/depth-camera-d435/

pose $\mathbf{P}_d$ with the right end-effector. In the control law (5.19) gain matrices $\bar{B}_d$ and $\bar{K}_d$ have a corresponding angular and linear part,

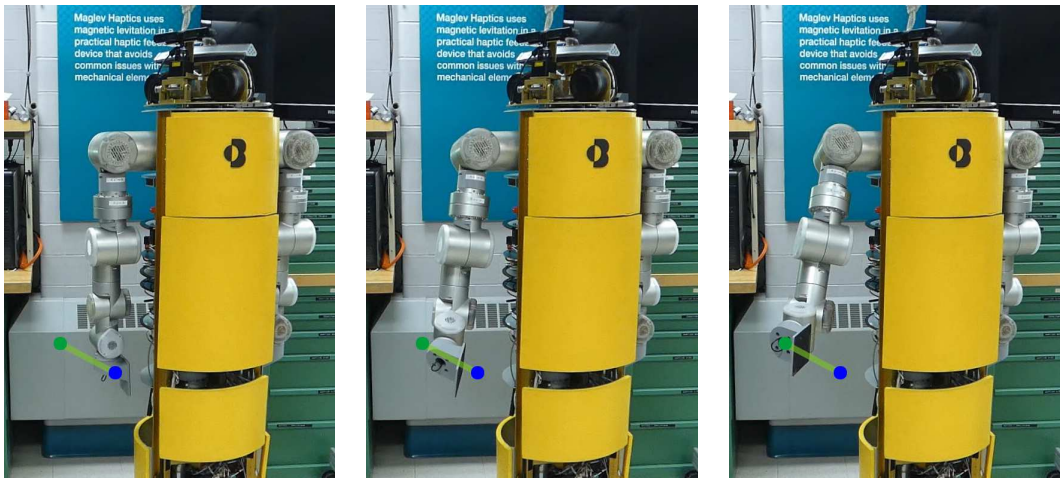
$$\bar{B}_d = \begin{bmatrix} b_{tran} & 0 \\ 0 & b_{rot} \end{bmatrix} \quad \text{and} \quad \bar{K}_d = \begin{bmatrix} k_{tran} & 0 \\ 0 & k_{rot} \end{bmatrix}. \tag{5.39}$$

The parameters $b_{tran} \in \mathbb{R}^{3\times3}$ and $b_{rot} \in \mathbb{R}^{3\times3}$ define the desired Cartesian damping applied to the three translation and rotational axis, respectively. The parameters $k_{tran} \in \mathbb{R}^{3\times3}$ and $k_{rot} \in \mathbb{R}^{3\times3}$ defined the desired Cartesian stiffness applied to the three translation and rotational axis, respectively. The control parameters and gains are set according to Table 5.1. For this experiment

	Experiment: 1
$\bar{B}_d$	$b_i = 2\sqrt{k_i m_i}$
k_{tran}	diag $([500, 500, 800])$
k_{rot}	diag $([50, 50, 50])$
$K_{q_a,p}$	diag $([1, 1, 1, 1, 1, 1])$
$K_{q_a,d}$	diag $([0.01, 0.01, 0.01, 0.01, 0.01, 0.01])$

Table 5.1: Parameters and gains for pure motion control experiment 1.

the values of $\bar{B}_d$ are set to achieve critical damping behavior (*i.e.* $b_i = 2\sqrt{k_i m_i}$). We assume all motions are in free space and forgo the use of the force feedback F_{ext}. The screenshots in Fig 5.5 show the right end-effector motion from its initial configuration in Fig. 5.5 (a) to reaching the target end pose in Fig. 5.5 (e). The task was repeated ten times with the same



(a) t = 0s (b) t = 7s (c) t = 10s

Figure 5.5: Screenshots of the motion control experiment tracking a minimum jerk trajectory between two poses of the right end-effector. The trajectory (green line) is realized in under 10s. The final translation and orientation errors are 0.007 m and 0.0277 rad (1.59 deg), respectively.

parameters. The mean linear and angular position trajectory of the right end-effector are shown

in Fig. 5.6 (a) and (b), respectively. The end-effector translation trajectory had a total length of 0.27 m with a maximum linear velocity of 0.6 m/s and is realized in less than 6 seconds. The end-effector orientation undergoes a large orientation change (*i.e.*, 90° in the Y-axis) throughout the trajectory. As shown in the time series plots of the tracking error in Fig. 5.6 (c) and (d), through

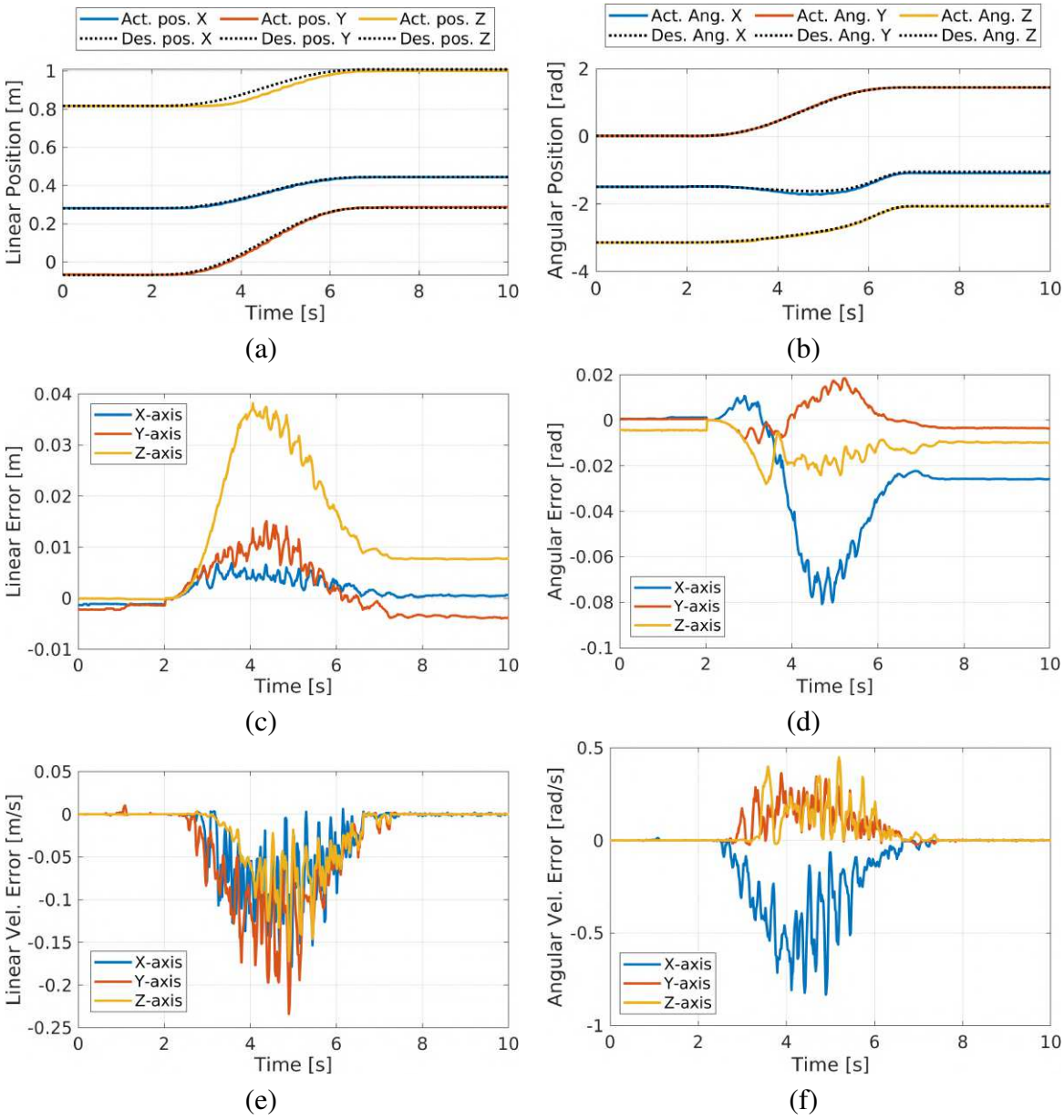


Figure 5.6: Experiment 1: The translation and angular right end-effector trajectory tracking performance.

the ten trials, the translation error is bounded within ± 0.01 m and the orientation error within ± 0.1 radians (5.78 degrees). Even with the higher stiffness in the z-axis (vertical), this coordinate had the biggest transient and steady-state error. A potential reason is that the controller is not fully compensating for gravity effects. Further, there could be unmodeled joint dynamics such as

stiction in the joints that contribute the most to the vertical movement (*i.e.* shoulder and elbow joints).

5.4.2.2 Experiment 2: Upper Body Posture Control

In this experiment, both the left and right arms are active. The task of the controller is to track a fixed end-effector pose with respect to the inertial frame $\{I\}$ while the ballbot is dynamically balancing and experiencing external forces from a human. The station-keeping controller is enabled to track a fixed ball position. The control parameters are set according to Table 5.2. The screenshots in Fig. 5.7 show how even though the ballbot is being pushed and pulled around

	Experiment: 2
$\bar{B}_d$	$b_i = 2\sqrt{k_i m_i}$
k_{tran}	diag $([800, 800, 600])$
k_{rot}	diag $([70, 70, 50])$
$K_{q_a,p}$	diag $([5, 5, 5, 5, 5, 5])$
$K_{q_a,d}$	diag $([0.01, 0.01, 0.01, 0.01, 0.01, 0.01])$

Table 5.2: Parameters and gains for pure motion control experiment 2.

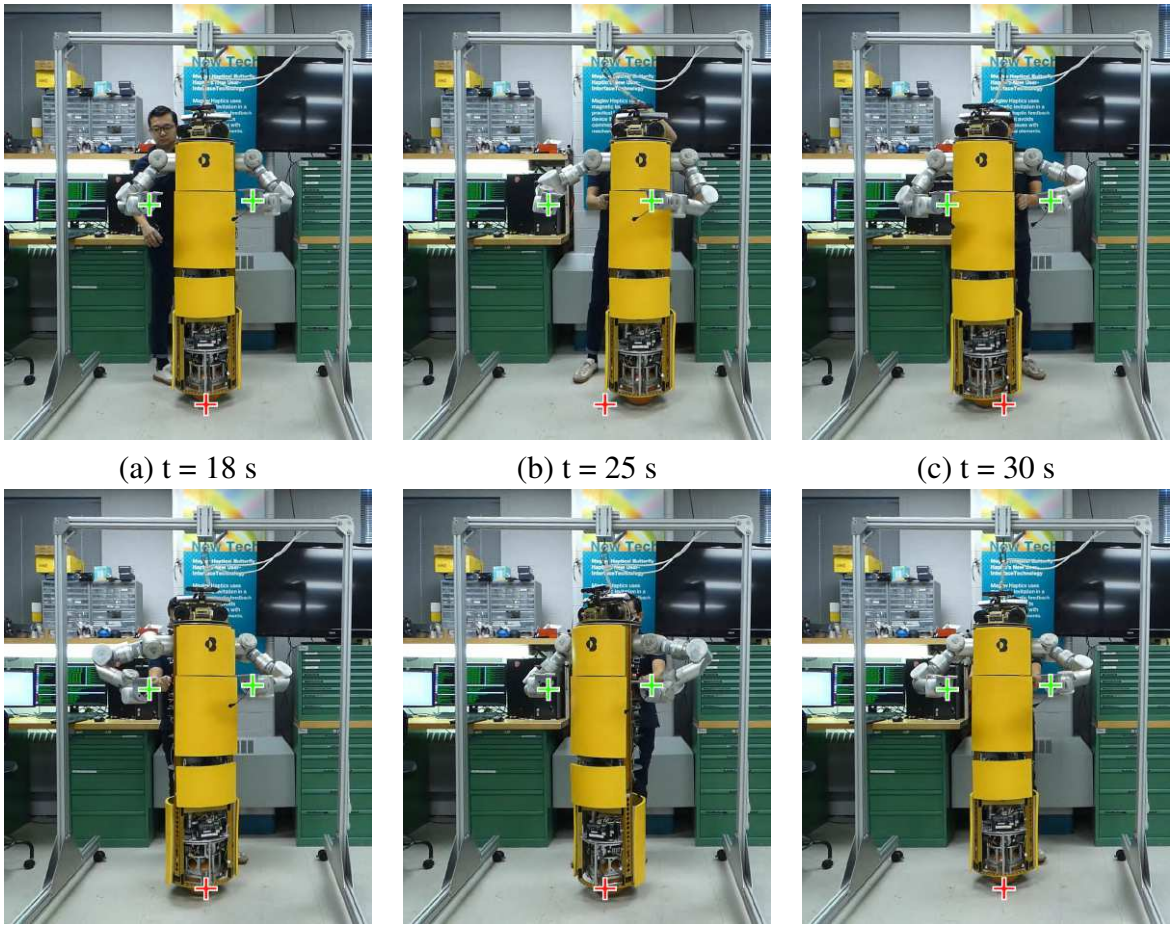
by the human, the controller can track the desired end-effector pose. The task space controller generates torque commands for both arms to control their internal configuration. The kinematic redundancy of the arm configuration allows compensating for both the translation and orientation error introduced. As shown in the time series plot in Fig. 5.8, the ball position is displaced ± 0.2 m on both directions. Despite this large displacement, the task space controller can track the end-effectors' positions in all directions within ± 0.05 m (*i.e.*, 25% of the introduced error). When the ballbot is yawed $\pm 30^\circ$ at 32 seconds, the tracking performances for the y-coordinate deteriorates to tracking within ± 0.1 m. The end-effector orientation error is kept within $\pm 4^\circ$ for the x and y-axis, as shown Fig. 5.8 (e) and (f). As expected, the orientation error in the Z-axis is greater because the corresponding stiffness gain is set to be lower by choice. The results of the posture control experiments can be viewed in the Video C.5.

5.5 Applications

Three different real-world applications have been tested on the CMU ballbot hardware that highlights the capabilities of the proposed control framework. These demonstrations show the potential of integrating a higher level of planning and reasoning to the task space impedance controller.

5.5.1 Balancing a Wine Glass while Disturbed

One practical demonstration of the controller's ability to track a desired end-effector pose is to balance a filled wine glass on its hand while dynamically balancing and experiencing external disturbances. Screenshots of this demo are show in Fig. 5.9. Note that the wine glass is not



(a) t = 18 s (b) t = 25 s (c) t = 30 s

(d) t = 32 s (e) t = 36 s (f) t = 44s

Figure 5.7: Screeshots of the motion control experiment tracking a desired pose for the right and left EE while the ballbot is dynamically balancing and external forces are applied (green X: desired EE pose, red cross: original ball position). In the first 12 seconds (a)-(c) the ballbot is experiencing a translation disturbance. In the last 12 seconds (d)-(f) the ballbot is being rotate around its central axis.

permanently attached to the ballbot's end-effector. The demonstration of the ballbot balancing a "wine glass" can be found in Video C.5.

5.5.2 Moving Cup in a Circle

The ballbot should stay in one place while the end-effector tracks the desired trajectory. In this demo, the ballbot tracks a desired ball position on the floor while simultaneously tracking a pose trajectory for the right end-effector and a fixed pose for the left end-effector. The right end-effector tracks a circular motion while keeping the desired orientation. To demonstrate it does so smoothly a cup of water is balanced on top of the end-effector throughout the motion. Screenshots of the motion are shown in Fig. 5.10. The entire motion can be viewed in the Video C.6.

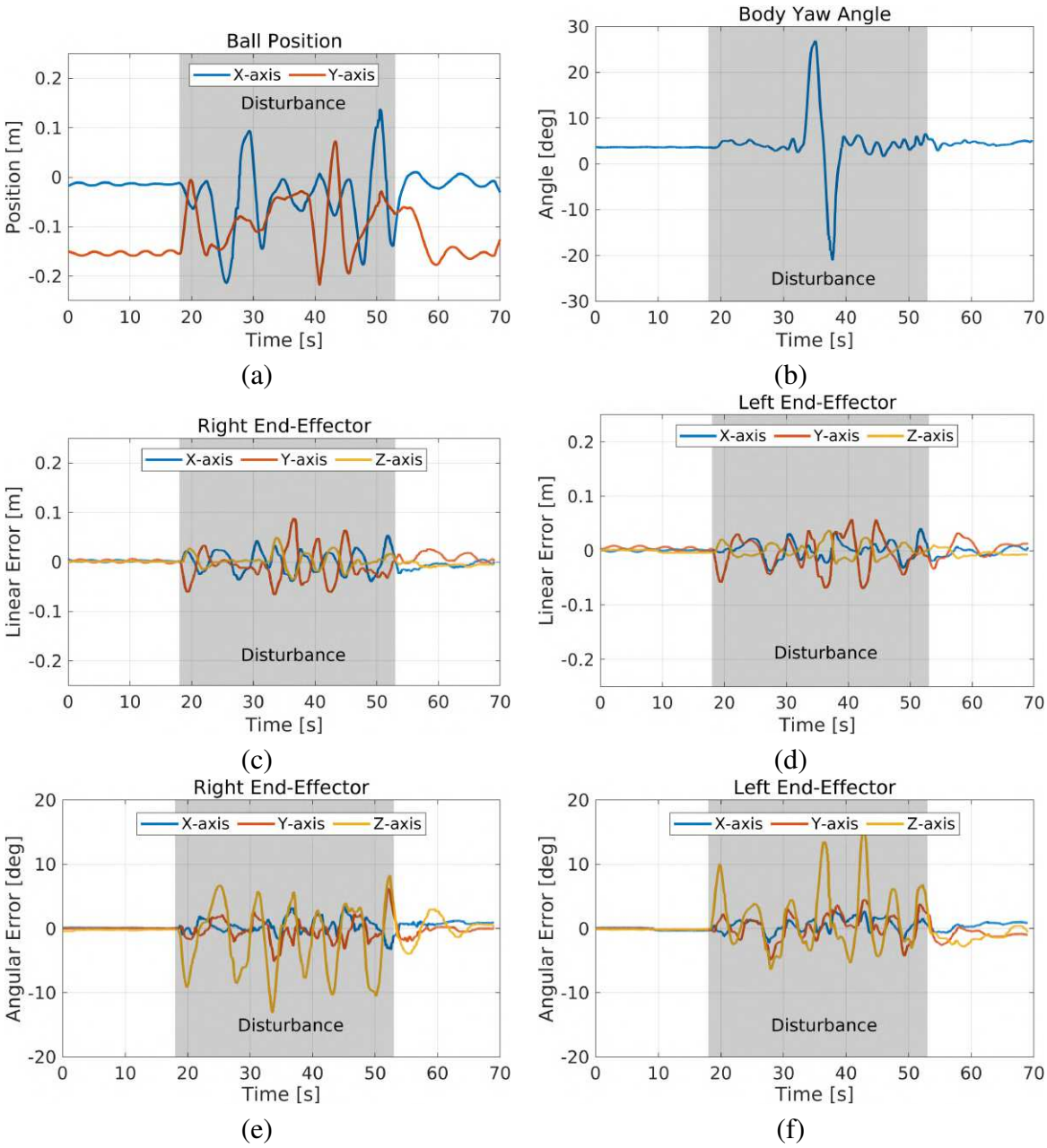
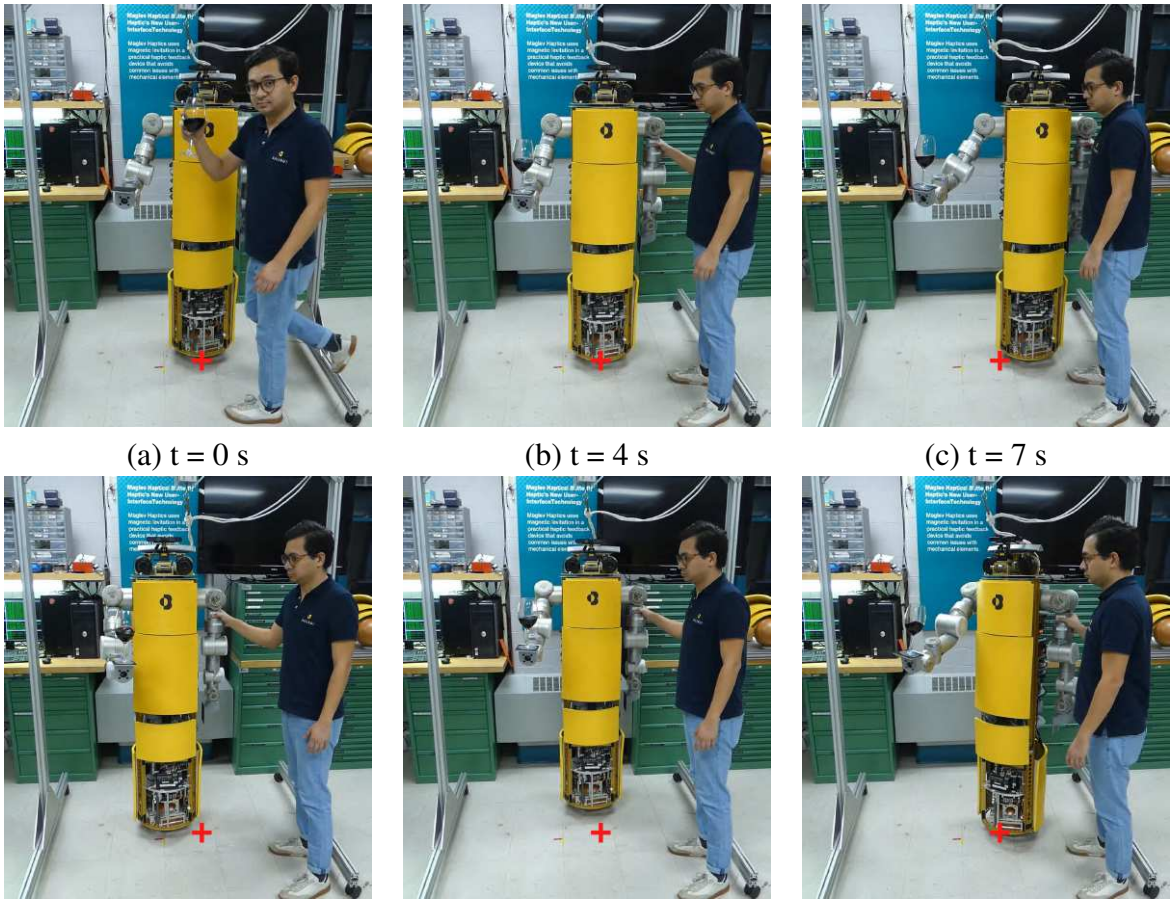


Figure 5.8: State evolution (a) the ball position on the floor, (b) the body yaw angle, (c) and (d) the right and left end-effector linear position error, (e) and (f) the right and left end-effector angular position error, during the upper body postural control experiment.

5.5.3 Picking up a Box

This task demonstrates the manipulation of an object in 6D-space with dual-arm manipulators like the CMU ballbot. The box is of an unknown dimension to the ballbot but is bounded. Instead of defining two end-effector trajectories (*i.e.*, one for each hand), we define a pose trajectory for a point between both hands and a desired separation between the hands. The surface of the



(a) t = 0 s (b) t = 4 s (c) t = 7 s

(d) t = 10 s (e) t = 16 s (f) t = 20 s

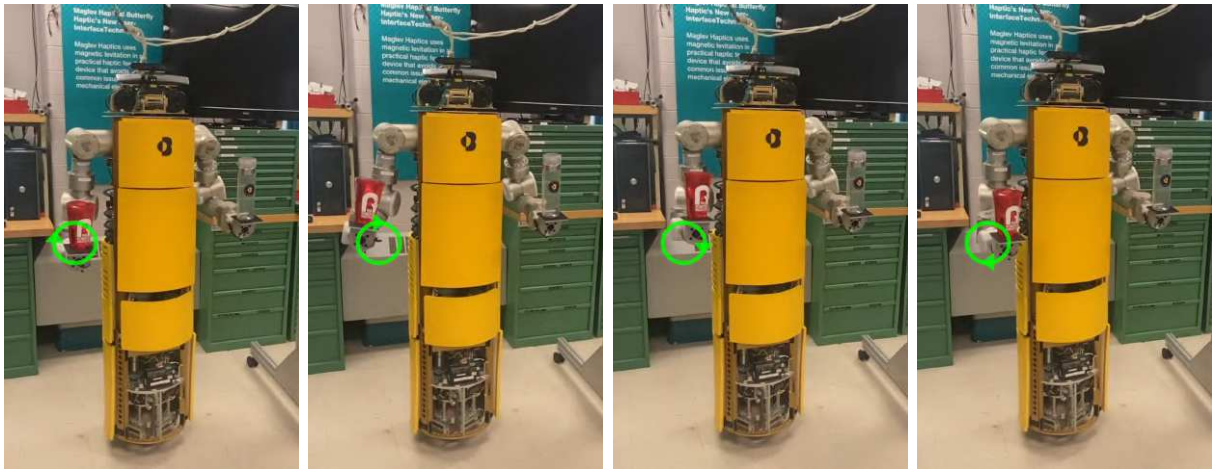
Figure 5.9: Screenshots of the application balancing a wine glass on the right end-effector while the ballbot is dynamically balancing and external forces are applied (red +: original ball position).

paddle hands are constrained to mirror each other throughout the entire motion. Screenshots of the ballbot picking up a box are shown in Fig. 5.11.

5.6 Discussion

This chapter presented a framework for controlling a dynamically stable mobile robot with redundant arms, like the ballbot with 7-DOF arms. The framework decomposes controlling the lower body, *i.e.*, body and ball, separately from the upper body, *i.e.*, the pair of arms. The lower body controller compensates for the motion of the arms. The upper body controller reacts to the motion of the lower body. This approach introduces pre-defined levels of hierarchy in the controller definition. The balancing controller has the highest priority in the control hierarchy. The task space controller developed enables *motion, force*, and *compliance control* of the arms' end-effectors. The performance of the controller was experimentally evaluated. Further, different real demonstrations of the task space control framework in action were presented.

Although this framework was shown to work on the CMU ballbot platform and perform



(a) t = 9s (b) t = 9.5s (d) t = 10 s (e) t = 10.5 s

Figure 5.10: Screenshots of the application moving a cup of water in a circular motion with the right end-effector while tracking a fixed left end-effector pose.

(a) t = 0s (b) t = 4s (c) t = 22s

Figure 5.11: Application: Picking up a box with both hands.

different demonstrations, it has several limitations. The biggest limitation of the framework is that all manipulation tasks are realized with a stationary ball position. That is, there is no consideration of combined locomotion and manipulation tasks. It is non-trivial to plan loco-manipulation tasks. First, a locomotion task has to be planned with stationary arms; then, once the ball position is stationary, a manipulation task can be planned. This framework does not leverage the body dynamics to perform manipulation (*e.g.* leaning forward to push open a door). To exploit the system's dynamics, careful coordination of the upper and lower body has to be planned. The following chapter presents a whole-body optimal planning and control method to solve the framework's limitation.

Chapter 6

Whole-Body Planning and Control

In Chapter 3 the problem of controlling dynamically stable shape-accelerating mobile manipulators, like the ballbot with arms, was decomposed into controlling the lower body, *i.e.*, body and ball, separately from the upper body, *i.e.* the pair of arms. The lower body controller reacted to the motion of the arms. This approach introduces pre-defined levels of hierarchies in the controller definition, thus limiting the system's full capabilities. To exploit the dynamics of the system, careful coordination between the upper and lower body motion is required. Solutions to this problem include search or optimization-based approaches that use either a complex, full-body nonlinear dynamic or a highly simplified robot model to generate motion plans.

This chapter presents a planning and control framework to generate dynamic, whole-body motions by considering both balancing and end-effector tasks in the same planning stage and over the same time horizon for a dynamically balancing mobile manipulator. The planner leverages the *centroidal dynamics* of the system. The framework first solves a trajectory optimization problem offline. It later uses the same Nonlinear Problem (NLP) with a shorter time horizon in a model predictive control (MPC) context to execute the motion. Balancing constraints in the optimization are formulated in terms of the *centroidal momentum* instead of other approaches like ZMP or angular velocity that are more commonly used. The proposed framework is evaluated in simulation and on CMU ballbot showing it can generate dynamic whole-body motion plans.

Parts of this chapter's contents appeared in [78].

6.1 Background

Whole-body planners and controllers have been extensively studied in legged humanoids [11, 73, 79, 80, 81], and quadruped robots [82, 83, 84], but scarcely for dynamically stable wheeled robots (e.g., Segway and ballbot bases). Kinematics-based controllers, like the one presented in Chapter 3, are most effective for slow upper-body motions. Considering the dynamics of the system can enable more agile and dynamic motions. Most existing dynamic-based algorithms use simplified models such as the Linear Inverted Pendulum (LIP) model that fail to exploit the whole body dynamic capabilities of the platform in use [4, 40, 85]. This model assumes that angular momentum is constant, which is not valid for motions requiring fast arm swinging. On the other hand, dynamic motion planning can be done by formulating a trajectory optimization prob-

lem with the full-body nonlinear dynamics of the system [86]. This method can produce smooth trajectories, but due to the complexity of the full-body nonlinear dynamics, these optimizations can take an excessively long time to run. This thesis instead represents the dynamics constraint of the ballbot by its *centroidal dynamics*, since postural balance can be defined in terms of centroidal momentum [87]. This method is a balance between the two extremes and has become a popular approach to plan and control legged humanoid and quadruped robots [11, 79]. When combined with full-body kinematics, the simple centroidal dynamics capture enough information to generate dynamic motions that leverage multiple limbs. Fig. 6.1 highlights the benefits and drawbacks of the different dynamic models.

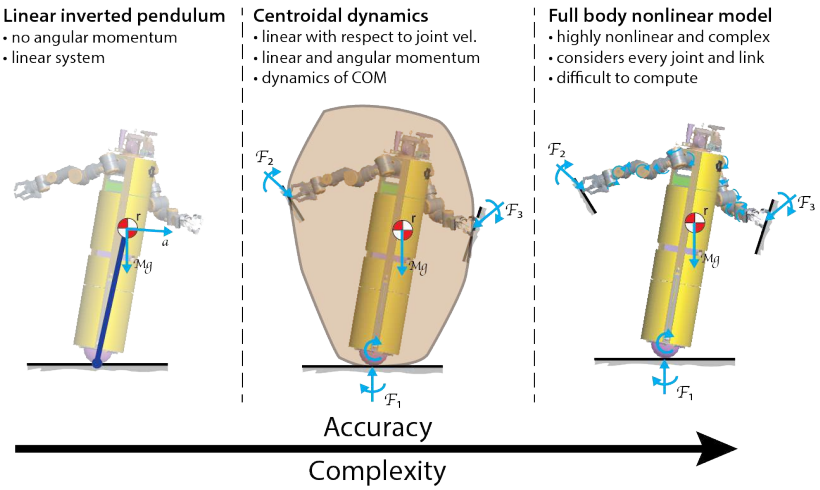


Figure 6.1: Overview of typical dynamical models used, ranging from a simple Linear Inverted Pendulum Model, to centroidal dynamics, to complex nonlinear full body dynamic models. In this work, we use the middle option of centroidal dynamics. (Inspired from [11])

Until recently, most balance control methods have attempted to maintain balance by controlling only the linear motion of the robot. An interesting departure from this are *momentum-based-balance controllers* [25]. These approaches control the spatial momentum's linear and angular components to perform whole-body motions.

In [25] a whole-body momentum-based controller for humanoid robots on non-level ground is presented. The controller regulates the linear and angular momentum by solving an optimization problem to find whole-body motion. It gives higher priority to linear momentum over angular momentum. This thesis implements a similar approach when both cannot be simultaneously attained.

In [88] a whole-body controller for a torque-controlled humanoid robot balancing on top of a two-wheel balancing platform is presented. The controller is formulated as a quadratic optimization problem (QP) to generate joint torques that satisfy the whole-body dynamics constraint. The quadratic cost function minimizes the error between the system's desired and actual linear and angular momentum. However, the issue of how to set the desired angular momentum for more complex motions such as performing locomotion and manipulation simultaneously was not fully explored.

Complex whole-body motions can be generated offline using trajectory optimization. In [89] a non-convex optimization problem is formulated that adopts centroidal dynamics to generate feasible trajectories for large-step up motions with a humanoid robot. The framework presented in this Chapter is inspired from [11], which combines the simple centroidal dynamics with a full kinematic model to generate whole-body motions. The work in this thesis differentiates in that the optimization problem is simplified by not including the complementary constraint for contact implicit optimization. This is possible because wheeled robots, like the ballbot, do not need to make and break contacts between the feet and floor to locomote. Also, instead of using a specialized QP for stabilizing dynamic locomotion [90], the same optimization problem formulation is reused and solved with a shorter horizon time inside an MPC framework. To decrease solving time, the solution of the offline optimization is used to warm-start the online MPC.

6.2 System Model

This section presents the dynamic model of the CMU ballbot with a pair of 7-DOF arms. The CMU ballbot is a human-size robot that balances on a ball. It has a pair of 7-DOF torque-controllable arms mounted onto the body. The ball is actuated using a four-motor inverse mouse-ball drive mechanism (IMBD). A pair of actuated opposing rollers drive the ball in each of the two orthogonal motion directions on the floor. This mechanism achieves omnidirectional motion by moving the ball in any direction. The IMBD 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). Another DC servomotor actuates this yaw degree of freedom. A slip ring assembly enables unlimited yaw rotation of the body. The model makes the following assumptions:

1. There is no slip between the ball and the floor.
2. The ball height relative to the floor remains constant, *i.e.*, the ball is always in contact with the floor.
3. There is no yaw motion between the floor and the ball.

6.2.1 3D Dynamics

The configuration of the robot is described by $n_b + \sum^i n_{a_i}$ DOF, where n_b is the number of DOF of the mobile base and n_{a_i} is the number of DOF of the i^{th} manipulators attached. The motion of the ballbot base can be described by $\mathbf{q}_b = [\mathbf{P}_S,\, \boldsymbol{\phi}]^T \in \mathbb{R}^{n_b}$ where $n_b = 5$. $\mathbf{P}_S = [p_x,\, p_y]^T \in \mathbb{R}^2$ is the 2D position of the ball in the horizontal plane with respect to the inertial frame $\{I\}$, $\boldsymbol{\phi} = [\phi_x,\, \phi_y, \phi_z]^T \in \mathbb{R}^3$ is a vector of Euler angles representing the lean angle of the body with respect to the gravity vector and the body yaw around the vertical axis. The 7-DOF of the left and right arms are represented by $\mathbf{q}_{a_L} \in \mathbb{R}^{n_a}$ and $\mathbf{q}_{a_R} \in \mathbb{R}^{n_a}$. These notations are shown in

Fig. 6.2. For the CMU ballbot the set of generalized coordinates

$$\mathbf{q} = \begin{bmatrix} \mathbf{P}_S \\ \phi \\ \mathbf{q}_{a_L} \\ \mathbf{q}_{a_R} \end{bmatrix} \text{ and } \dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{P}}_S \\ \dot{\phi} \\ \dot{\mathbf{q}}_{a_L} \\ \dot{\mathbf{q}}_{a_R} \end{bmatrix} \tag{6.1}$$

is defined. The system has a total of $n = 19$ DOF ($n_b = 5, n_{a_L} = 7, n_{a_R} = 7$). Grasp-planning is outside the scope of this work and hence the DOF of the hand fingers are not considered. Instead, we only consider the left and right arm end-effector pose $\mathbf{p}_{EE,L} \in SE(3)$ and $\mathbf{p}_{EE,R} \in SE(3)$, respectively. We also define $\mathbf{r} \in \mathbb{R}^3$ the robot's COM position with respect to $\{I\}$.

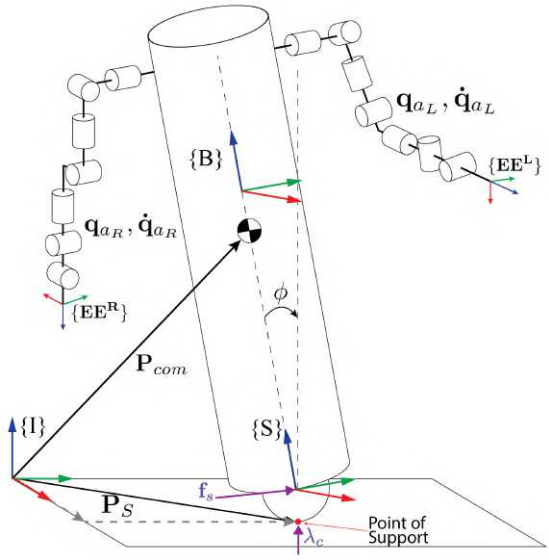


Figure 6.2: Schematic of the CMU ballbot generalized coordinate representation. $\{I\}, \{S\}, \{B\}$ are the inertial frame, the ball (sphere) frame and the body frame. $\{EE^R\}$ and $\{EE^L\}$ are right and left end-effector frames.

Considering the generalized coordinates defined in (6.1) the Euler-Lagrange Equation of Motion (EOM) are as follow:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{S}\boldsymbol{\tau} + \boldsymbol{J}_c^T \boldsymbol{\lambda}_c, \tag{6.2}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ is the matrix composed of the sum of Coriolis and centrifugal forces, and $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^n$ is the gravity force vector, $\boldsymbol{\lambda}_c$ is the floor reaction force, $\boldsymbol{J}_c$ is the corresponding Jacobian. $\boldsymbol{\tau} = [\mathbf{f}_s, \tau_{\phi_\mathbf{z}}, \boldsymbol{\tau}_{\mathbf{a,L}}^{\mathbf{T}}, \boldsymbol{\tau}_{\mathbf{a,R}}^{\mathbf{T}}]^{\mathbf{T}} \in \mathbb{R}^{\mathbf{n}_\tau}$ is the vector of generalized forces and torque inputs. The torque exerted by the IMBD at the center of the ball is related to a linear force at the point of contact between the ball and the floor by

$$\mathbf{f}_s = \begin{bmatrix} f_s^x \\ f_s^y \end{bmatrix} = \begin{bmatrix} r_b \cdot \tau_b^x \\ r_b \cdot \tau_b^y \end{bmatrix}, \tag{6.3}$$

where r_b is the radius of the ball and the τ_b^x, τ_b^y are the torques exerted by the IMBD in the X and Y axis respectively. $\tau_{\phi_\mathbf{z}}$ is the torque applied on the yaw degree of freedom. The joint torque of the manipulators are described by the vectors $\boldsymbol{\tau}_{a,L} \in \mathbb{R}^{n_{a_L}}$ and $\boldsymbol{\tau}_{a,R} \in \mathbb{R}^{n_{a_R}}$. The system has a total of sixteen control inputs, *i.e.*, $n_\tau = 17$. The actuation selection $\mathbf{S} \in \mathbb{R}^{n \times (n_\tau)}$ separates the $n_c = n - n_f$ controlled joints from the $n_f = 2$ unactuated body lean angle joints ϕ_x and ϕ_y. Since the number of degrees of freedom of the robot is larger than the number of independent control inputs, the system is underactuated.

6.2.2 Centroidal Dynamics

The centroidal dynamics are the dynamics of a robot system projected at its COM [87]. They describe the evolution of the angular momentum and the COM position [11]. The centroidal momentum vector $\mathbf{h} \in \mathbb{R}^{6\times1}$ composed of the linear momentum $\mathbf{l} \in \mathbb{R}^{3\times1}$ and angular momentum $\mathbf{k} \in \mathbb{R}^{3\times1}$ is linearly related to the generalized joint velocities vector $\dot{\mathbf{q}}$ by

$$\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} \mathbf{k} \\ \mathbf{l} \end{bmatrix} = \mathbf{A}(\mathbf{q})\dot{\mathbf{q}}, \tag{6.4}$$

where $\mathbf{A} \in \mathbb{R}^{6\times n}$ is the centroidal momentum matrix (CMM). Taking the time derivative of (6.4) results in the second order *centroidal dynamics*

$$\dot{\mathbf{h}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{A}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}. \tag{6.5}$$

The rate of centroidal linear and angular momentum $\dot{\mathbf{h}}(\mathbf{q}, \dot{\mathbf{q}}) = [\dot{\mathbf{k}}, \dot{\mathbf{l}}]^T$, computed from the robot's joint angles and velocities, equals the total wrench generated by the external contacts and the gravitational forces:

$$\dot{\mathbf{k}} = \mathbf{m}\ddot{\mathbf{r}} = \sum_j \mathbf{F}_j + \mathbf{mg} \tag{6.6}$$

$$\dot{\mathbf{l}}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_j (\mathbf{c}_j - \mathbf{r}) \times \mathbf{F}_j + \tau_j, \tag{6.7}$$

where $\mathbf{m}$ is the total mass of the robot, $\mathbf{r} \in \mathbb{R}^3$ is the COM position, $\mathbf{F}_j \in \mathbb{R}^3$ is the external contact force at j^{th} contact point, $\mathbf{c}_j \in \mathbb{R}^3$ is the position of the j^{th} contact point, $\tau_j \in \mathbb{R}^3$ is the torque at the j^{th} contact point, and $g \in \mathbb{R}^3$ is the gravitational acceleration. The CMM has been shown to be useful when generating dynamic motions of multiple limbs to maintain balance [87]. Thus, in this thesis work balancing is defined in terms of the linear and angular momentum. For balancing maintenance it is desired that $\mathbf{k}$ and $\mathbf{l}$ be zero and for $\mathbf{P}_{com}$ to be above the point of support. This will form the basis of the optimization problem presented in Sec. 6.3.

6.3 Optimization

6.3.1 Trajectory Optimization

To compute the feasible motion plan that includes the centroidal momentum trajectory and joint trajectories, a nonlinear optimization problem (NLP) is formulated. The centroidal dynamics

and a full kinematic model are used to enforce dynamic and geometric constraints [11]. A direct collocation method [91, 92] to transcribe the continuous-time dynamics in (6.4) and (6.5) to their discrete form is implemented. All time-varying quantities are sampled at N knot points. The nonlinear constraint optimization problem minimizes the cost function

$$\min_{\substack{\mathbf{q}[k],\dot{\mathbf{q}}[k],\ddot{\mathbf{q}}[k],\\ \mathbf{r}[k],\dot{\mathbf{r}}[k],\ddot{\mathbf{r}}[k],\\ \mathbf{h}[k],\dot{\mathbf{h}}[k],\\ \mathbf{F_j}[k],\tau_\mathbf{j}[k]}} \sum_{k=0}^{N} \Big(\big|\mathbf{P}_{EE,i}[k] - \mathbf{P}^d_{EE,i}[k]\big|^2_{\mathbf{Q}_{EE,i}} + \big|\mathbf{e}_{oEE,i}[k]\big|^2_{\mathbf{Q}_{oEE,i}} + \big|\mathbf{P}_S[k] - \mathbf{P}^d_S[k]\big|^2_{\mathbf{Q}_S} + \big|\mathbf{P}_S[k] - \mathbf{P}_{com}[k]\big|^2_{\mathbf{Q}_{com}} + \big|\dot{\mathbf{h}}[k]\big|^2_{\mathbf{Q}_{\dot{\mathbf{h}}}} + \big|\ddot{\mathbf{q}}[k]\big|^2_{\mathbf{Q}_{\ddot{\mathbf{q}}}} \Big), \quad (6.8)$$

where $|x|^2_{\mathbf{Q}}$ is the abbreviation for the quadratic cost $x^T Q x$. The square bracket $[k]$ means the sampled value at the k^{th} knot point. The cost function (6.8) tries to minimize the sum of different tasks that the robot is required to perform. The semidefinite positive matrices $Q_{EE,i}$ and $Q_{oEE,i}$ weight the task of tracking a desired end-effector position and orientation for $i = L, R$ corresponding to the left and right end-effectors. The orientation error is computed using quaternion difference. The task of tracking the desired base position is weighted by the matrix Q_S. The balancing task is defined by the error tracking term between the COM position to be on top of the point of support and the term penalizing the centroidal momentum rate. The momentum penalizing term is weighted higher by $Q_{\dot{\mathbf{h}}}$ than the COM tracking term by Q_{com}. The regulation term penalizing joint acceleration in the cost function is included to provide numerical stability and is weighted by $Q_{\ddot{\mathbf{q}}}$.

The formulation of the NLP is flexible in the sense that different behaviors and tasks can be achieved with the same cost function by simply changing the weights on different terms. For example, if we care about tracking the right end-effector position more than the left end-effector, then we can set $Q_{EE,R} \geq Q_{EE,L}$.

The optimization constraints include the centroidal dynamics defined in (6.4) and (6.5) discretized at each knot point

$$m\ddot{\mathbf{r}}[k] = \sum_i \mathbf{F}_i[k] + m\mathbf{g}, \quad (6.9)$$

$$\dot{\mathbf{h}}[k] = \sum_i (\mathbf{c}_i[k] - \mathbf{r}[k]) \times \mathbf{F}_i[k] + \tau_j[k], \quad (6.10)$$

$$\mathbf{h}[k] = \mathbf{A}(\mathbf{q}[k])\dot{\mathbf{q}}[k]. \quad (6.11)$$

Note that in our case, there is only one contact point $\mathbf{c}_i$ between the ball and the floor. To enforce continuity between the discrete system state knot points $\mathbf{q}[k], \dot{\mathbf{q}}[k], \ddot{\mathbf{q}}[k], \mathbf{h}[k]$, and $\dot{\mathbf{h}}[k]$ we formulate equality constraints such that the change in state between two knot points is equal to the integral of the system dynamics. We approximate the integral using Euler integration. For numerical stability, this is implemented using backward-Euler integration. The collocation

constraints are

$$\mathbf{q}[k] - \mathbf{q}[k-1] = \dot{\mathbf{q}}[k]dt, \tag{6.12}$$
$$\dot{\mathbf{q}}[k] - \dot{\mathbf{q}}[k-1] = \ddot{\mathbf{q}}[k]dt, \tag{6.13}$$
$$\mathbf{h}[k] - \mathbf{h}[k-1] = \dot{\mathbf{h}}[k]dt, \tag{6.14}$$

where dt is the sample time between knot points. We approximate the COM position using a piecewise quadratic polynomial. Its time integration constraints are

$$\mathbf{r}[k] - \mathbf{r}[k-1] = \frac{\dot{\mathbf{r}}[k] + \dot{\mathbf{r}}[k-1]}{2}dt, \tag{6.15}$$
$$\dot{\mathbf{r}}[k] - \dot{\mathbf{r}}[k-1] = \ddot{\mathbf{r}}[k]dt. \tag{6.16}$$

To ensure the full kinematics of the robot are obeyed we implement the kinematic constraint that relates the robot joint configuration and COM position

$$\mathbf{r}[k] = \text{com}\left(\mathbf{q}[k]\right), \tag{6.17}$$

where $\text{com}(\mathbf{q})$ computes the corresponding COM position to a given robot joint configuration $\mathbf{q}$. Joint position, velocity, and acceleration limits are also enforced through the inequality constraints

$$\mathbf{q}_{lb} \leq \mathbf{q}[k] \leq \mathbf{q}_{ub}, \tag{6.18}$$
$$\dot{\mathbf{q}}_{lb} \leq \dot{\mathbf{q}}[k] \leq \dot{\mathbf{q}}_{ub}, \tag{6.19}$$
$$\ddot{\mathbf{q}}_{lb} \leq \ddot{\mathbf{q}}[k] \leq \ddot{\mathbf{q}}_{ub}. \tag{6.20}$$

Equality boundary constraints are also included to enforce initial and final robot states. The initial constraints for the joint states are set to

$$\mathbf{q}[0] = \mathbf{q}_0, \quad \dot{\mathbf{q}}[0] = \dot{\mathbf{q}}_0, \quad \text{and} \quad \ddot{\mathbf{q}}[0] = \ddot{\mathbf{q}}_0. \tag{6.21}$$

The final value of the centroidal momentum and centroidal momentum rate are constrained to be zero, so to ensure the robot is balancing at the end of the motion by

$$\mathbf{h}[N] = 0 \quad \text{and} \quad \dot{\mathbf{h}}[N] = 0. \tag{6.22}$$

6.3.2 Model Predictive Control

The NLP described in Section 6.3.1 outputs smooth end-to-end robot motion trajectories. However, this NLP has a long time horizon and takes several seconds to solve. Thus, it is only good to be solved offline. Further, executing this trajectory in an open-loop fashion on the robot may lead to poor performance due to model uncertainties and unmodeled dynamics. In this work, a model predictive control (MPC) is implemented using a similar NLP as the trajectory optimization problem. However, by shortening the time horizon $N \leq 10$ and using the output of the offline trajectory optimization to warm-start, we can reduce the solve time to be fast enough

($\approx 20Hz$) to be computed in an MPC fashion. The constraint on the final zero centroidal momentum is dropped, as this will generate undesirable stopping behaviors. The complete planning and control structure is depicted in Fig. 6.3.

Instead of feeding the output of the MPC directly to the robot motors, the MPC output is fed to a PD-PID cascading balancing controller. This ensures that the robot will maintain balance at all times. The inner PID loop running at 500 Hz maintains the ballbot balancing upright. It does so by tracking a desired lean angle by actuating the ball. The outer loop tracks the ball position on the floor and feeds lean-angle setpoints to the inner-loop controller. By tracking the body's lean angle we can indirectly track the ball position in the floor. The arms are controlled by a decentralized torque-impedance-based feedback controller with feedforward gravity and torque-sensing compensation, as shown in Eq. 6.23.

$$\tau_{des} = K_{P_\alpha} e_\alpha + K_{D_\alpha} \dot{e_\alpha} + g(\alpha, \dot{\alpha}) - \tau, \tag{6.23}$$

where $K_{P_\alpha}, K_{D_\alpha}$ are positive definite diagonal gain matrices, $g(\alpha, \dot{\alpha})$ is the gravity compensation term based on the dynamic model of the arm, $e_\alpha = \alpha_{des} - \alpha$ and $\dot{e}_\alpha = \dot{\alpha}_{des} - \dot{\alpha}$ are the joint position and velocity errors.

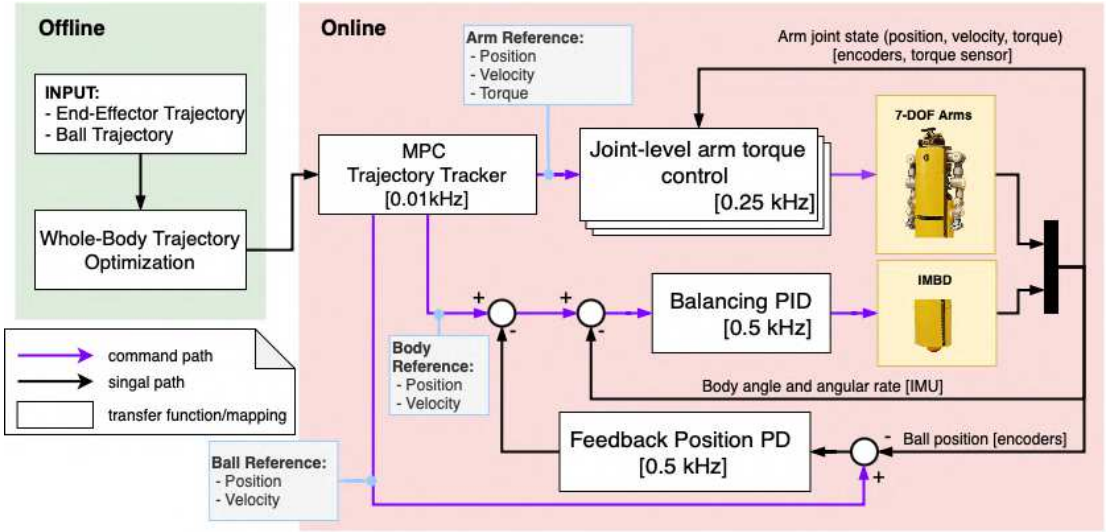


Figure 6.3: Block diagram of the implemented whole-body optimal control framework for ballbot using centroidal dynamics.

6.4 Experiments and Results

The proposed whole-body planning and control framework was tested in a variety of tasks in a dynamic simulation environment of the CMU ballbot and in the real hardware.

6.4.1 Implementation Details

All optimisation problems are solved using Ipopt [93] solver through CasADi [94] C++ interface. The robot's dynamic equations are obtained from the rigid body dynamic library Pinocchio [95], which provides a CasADi interface. For all the tasks tested, the offline trajectory optimization problem is solved with a long time horizon ($N \geq 40$ and $dt = 0.1$ s) to obtain a reference trajectory to follow. All offline optimization problems were run on a PC with an Intel Xeon(R) 2.80GHz quad-core CPU running Ubuntu 16.04.

The online optimization and controllers are implemented in C++ wrapped into ROS [38]. This optimization uses a short time horizon ($N \leq 10$) and runs online to track the motion plan in a Model Predictive Control fashion. The output solution of the trajectory optimization problem is sampled on every iteration to warm-start the online optimization problem.

The simulation test environment is written in C++ using the Pinocchio [95]. The evolution of the dynamics is obtained using an explicit Euler integration scheme. Pinocchio generates the model based on the Unified Robot Description Format (URDF[1]) model of the robot and the parameters of the robot obtained from system identification.

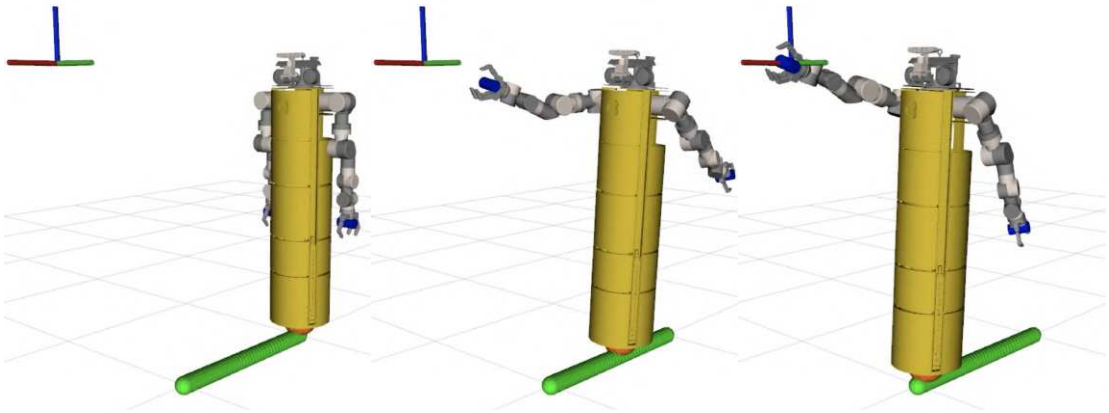
6.4.2 Simulation

This section presents the simulation results of the whole-body planning and control framework for the CMU ballbot with its pair of 7-DOF arms. The simulation results are shown in the supplemental Video C.7.

6.4.2.1 Experiment 1: Single End-Effector Pose Tracking

The primary task for a ballbot is to maintain balance, but with the addition of the 7-DoF arms manipulation tasks become important too. In this first experiment, we set a reference position and orientation for the right end-effector that is outside the arm's reach without moving the base, as shown in Fig. 6.4. A 5 kg payload is added to each hand to make the task more difficult. For this task we penalize the end-effector position and orientation error $\mathbf{Q}_{EE,r} = \text{diag}([100, 100, 100])$ and $\mathbf{Q}_{oEE,r} = \text{diag}([10, 10, 10])$. All other weights matrices $\mathbf{Q}_i$ are set to the identity. Without explicitly defining a task for the base motion, the framework synthesizes a combined motion for the arms and base such that the robot achieves its end-effector task while maintaining balance. The centroidal momentum trajectory is shown in Fig. 6.5. The planner outputs a motion that uses its free arm to balance by swinging it backward. However, this motion is not ideal when operating in a tight environment. The motion can be easily modified by simply increasing the weight $Q_{\ddot{\mathbf{q}}}$ on the cost of moving the joints of the left arm. Fig. 6.6 shows the results of achieving the same task but now using the body lean angle to maintain balance instead of the second arm. The left arm remains in its zero-configuration throughout the entire motion.

[1] http://wiki.ros.org/urdf



(a) t = 0 sec (b) t = 3 sec (c) t = 5 sec

Figure 6.4: Snapshots of tracking a desired end-effector pose with low weight cost on left arm joint acceleration.

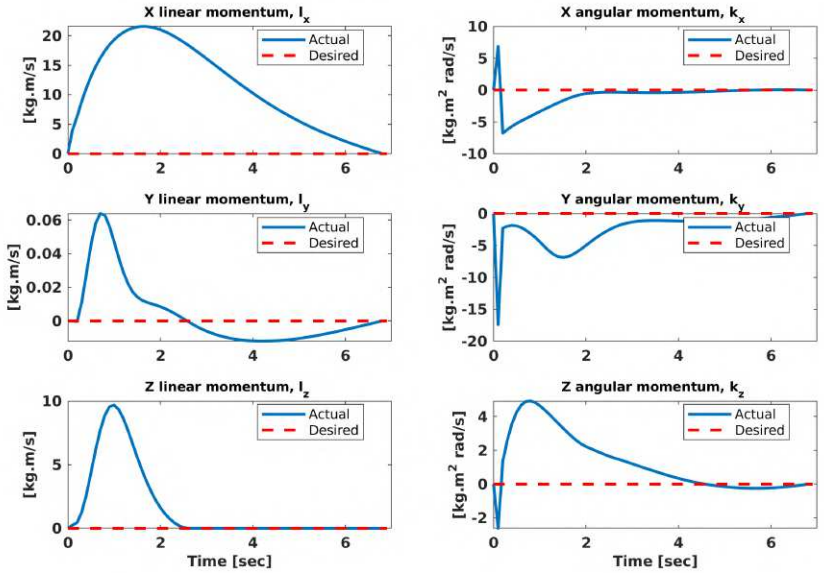
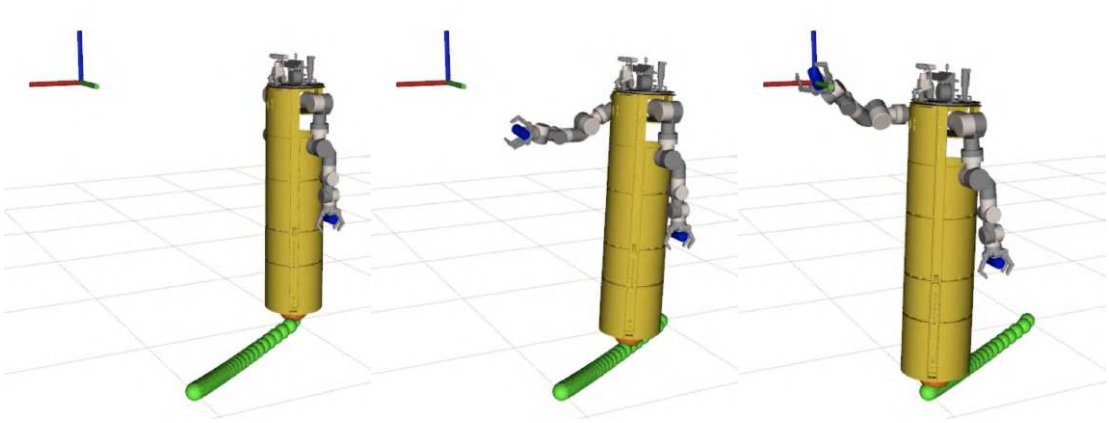


Figure 6.5: Centroidal momentum trajectory when tracking a single end-effector pose.

6.4.2.2 Experiment 2: Dual End-Effector Pose Tracking

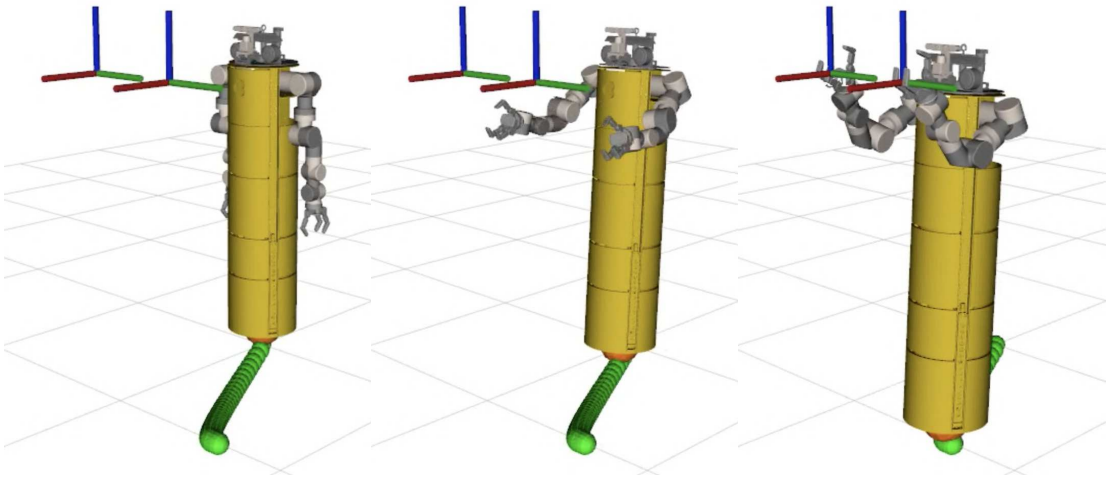
In this experiment, we control the desired position and orientation for both end-effectors as shown in Fig. 6.7. We set high weights on the position $Q_{EE,L} = Q_{EE,R} = \text{diag}\big([100, 100, 1000]\big)$ and orientation $Q_{oEE,L} = Q_{oEE,R} = \text{diag}\big([50, 50, 50]\big)$ error tracking terms. In Fig. 6.8 the end-effector position tracking error are shown. The linear and angular momentum trajectories are non-zero initially to realize the desired motion but quickly return to zero to stabilize the robot, as shown in Fig. 6.9. We set $Q_b = 0$ to give the optimization the freedom to find a motion to coordinate the base and arm motion to track the desired end-effector pose. As desired, the controller can successfully track the planned motion that uses the body lean to compensate for the COM



(a) t=0 sec (b) t=3 sec (c) t=5 sec

Figure 6.6: Snapshots of tracking a desired end-effector pose with large weight cost on left arm joint acceleration.

movement due to the arm motion. The offline plan was generated in 26 seconds. This behavior is very similar to that of humans when lifting heavy objects.



(a) t=0 sec (b) t=3 sec (c) t=6 sec

Figure 6.7: Snapshots of tracking desired position and orientation for both end-effectors. Note the use of the arms instead of the body lean to move towards the target location.

6.4.2.3 Experiment 3: Base Position Tracking

There may be scenarios where we are not interested in the end-effector pose and are only interested in the base tracking the desired position. This can be accomplished by setting $Q_{EE,i} = 0$ and setting a large value for $Q_b = \text{diag}\big([100, 100]\big)$. We test this by commanding the robot to a

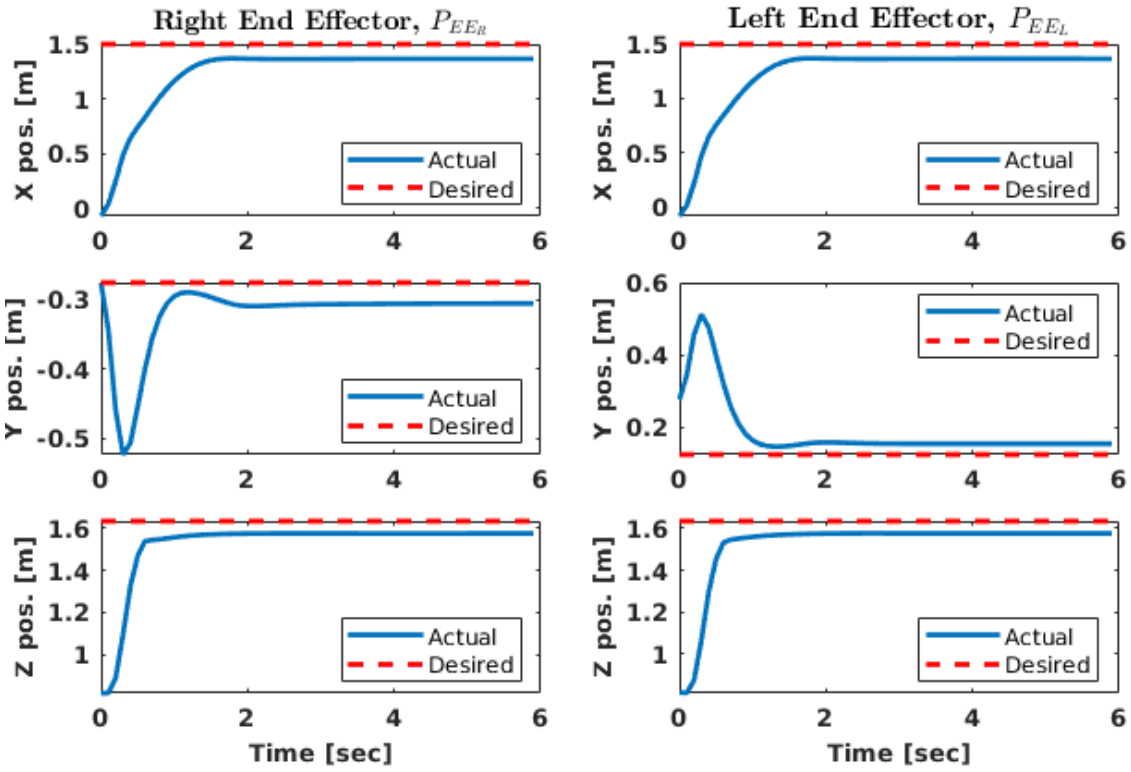


Figure 6.8: Right and left end-effector cartesian position with respect to inertial frame.

position $\mathbf{P}_B^d = [1,\ 1]^T$ m, as shown in Fig. 6.10. Without setting a high weight to penalize the arm joint accelerations, the trajectory optimization found an optimum motion in 21 seconds that primarily uses lean angle motion. Screenshots of the entire motion are shown in Fig. 6.11. Even without setting a high weight on the joint accelerations (*i.e.*, $Q_{\ddot{q}} \leq I$) minimum arm motion is used. This is desirable and expected since inducing a small lean angle alone produces enough momentum to realize the motion. Swinging the arms will produce undesirable large momentum changes. This result resembles those obtained from our differential flatness-based planner [34].

6.4.3 Hardware

In this section, we present the results obtained on the real CMU ballbot hardware.

6.4.3.1 Experiment 1: Nearby Right End-Effector Target

In this first experiment, the task is to position the right end-effector in a desired Cartesian position within the arm's reach without navigating. For this task we penalize the end-effector position error $\mathbf{Q}_{EE,r} = \text{diag}([150, 150, 150])$. All other weights matrices $\mathbf{Q}_i$ are set to the identity. The motion plan was generated offline in 21 seconds. Similar to before, the framework can generate and track a combined motion for the arm and base that attains the desired end-effector position. Snapshots of the entire motion running on the robot are shown in Fig. 6.12. The blue ball is

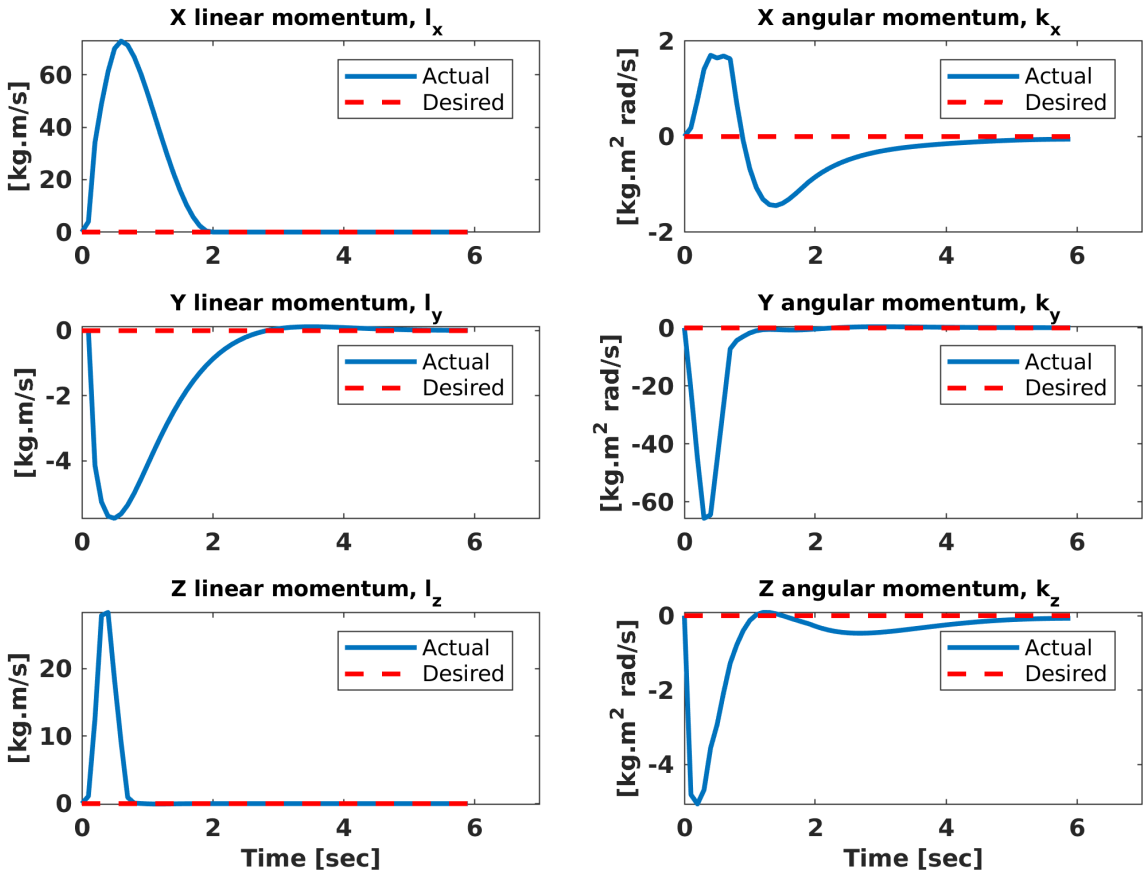


Figure 6.9: Linear and angular momentum evolution while tracking a desired position for both end-effectors

only used as a visual reference of the desired end-effector position, and it is not tracked online. Fig. 6.13 shows the trajectory of the end-effector throughout the entire motion. The controller does a good job at tracking the final end-effector position in the x-y direction (*i.e.* tracking error is < 0.01 m). However, in the vertical Z-direction the tracking error is larger (*i.e.* < 0.07 m). The error in the Z-axis could be caused by poor arm joint stiction and gravity models.

6.4.3.2 Experiment 2: Far Away Right End-Effector Target

In the next experiment, the task is to position the right end-effector in a desired Cartesian position that is outside the arm's reach without navigating, as shown in Fig. 6.14 (a). Again, the blue ball is only used as a visual reference for the target end-effector position and is not actively tracked. To reach the desired end-effector pose, the ballbot has to navigate towards the target. The end-effector position relative to the initial base position is $\mathbf{P}^{s}_{EE,r} = [1.54, 0.97, 1.21]$. This task is particularly hard because the right arm has to cross the mid-plane of the robot and navigate diagonally. For this task we penalize the end-effector position error $\mathbf{Q}_{EE,r} = \text{diag}([100, 100, 100])$.

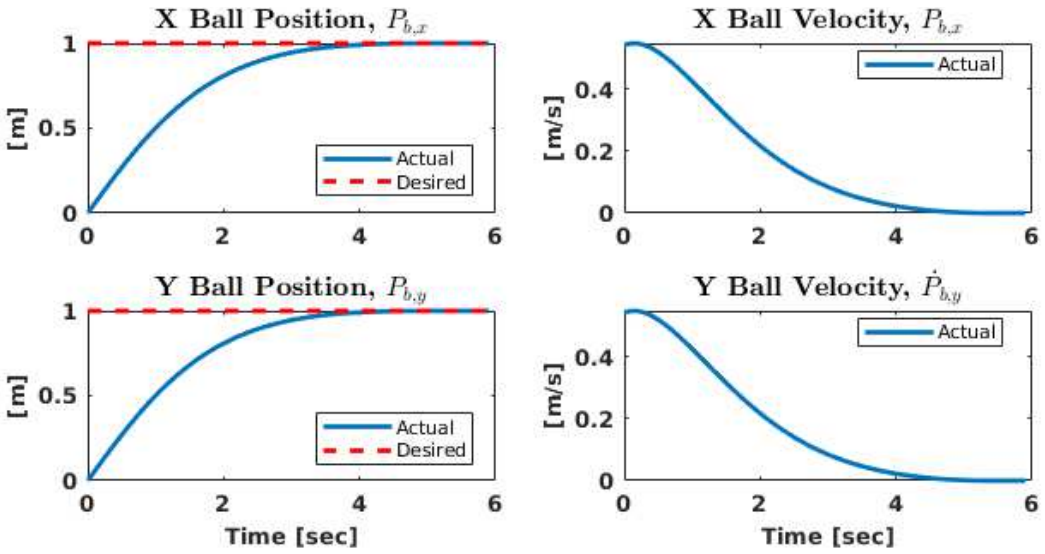
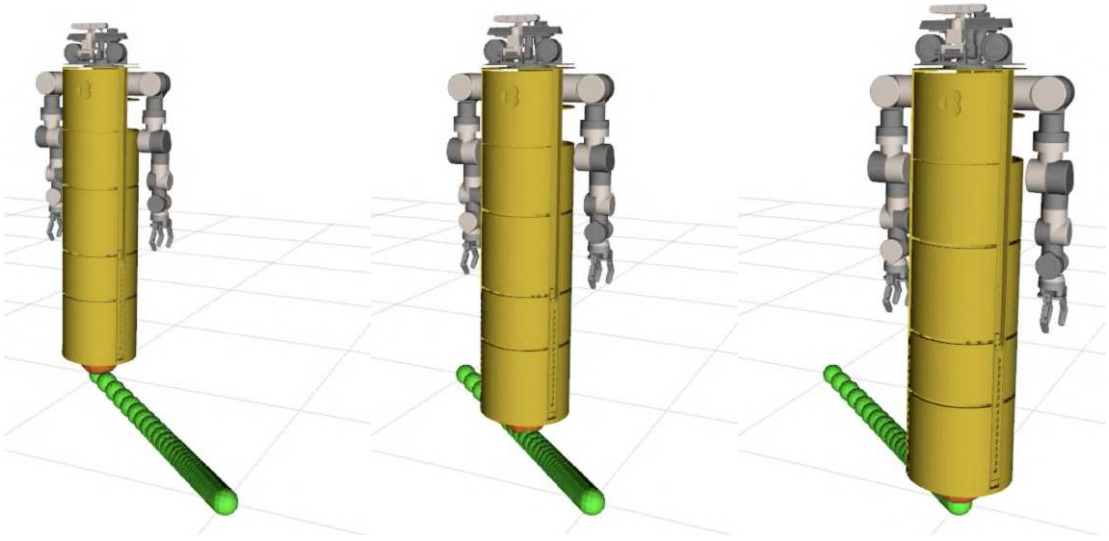


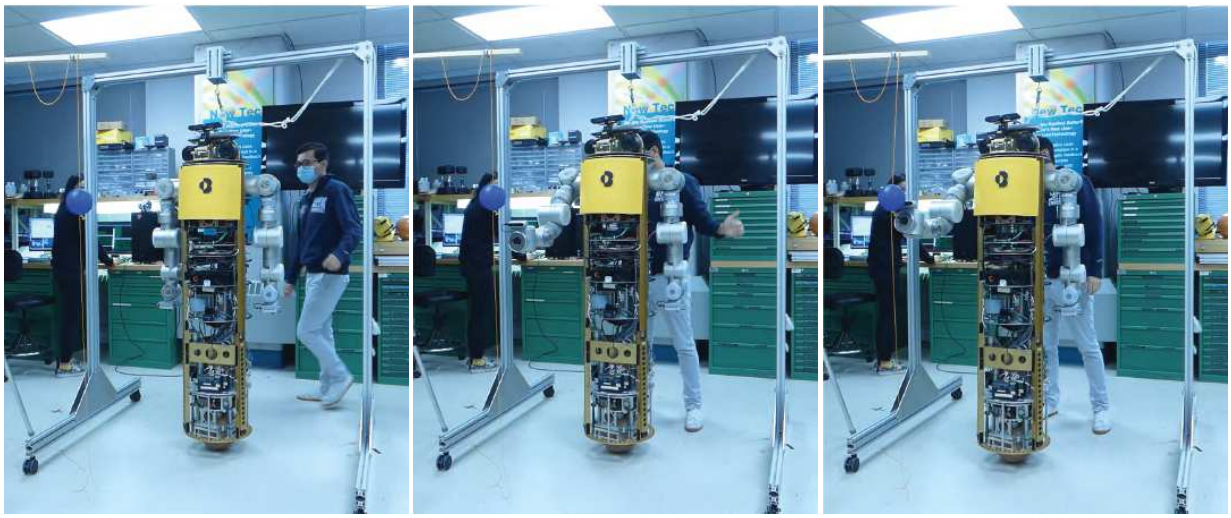
Figure 6.10: Ball position and velocity trajectories for the task of tracking a desired ball position



(a) t=0 sec (b) t=3 sec (c) t=6 sec

Figure 6.11: Snapshots of tracking desired base position. Despite arm joint acceleration not being penalized they are not used to generate forward momentum.

All other weights matrices $\mathbf{Q}_i$ are set to the identity. The motion plan was generated offline in 38 seconds. Snapshots of the entire motion are shown in Fig. 6.14. There is some oscillation in the ballbot's ball motion before it comes to rest at the target location. Once the ballbot reaches an equilibrium state, the end-effector settles to its desired position.



(a) t = 0 sec (b) t = 2 sec (c) t = 5 sec

Figure 6.12: Screenshots of tracking a nearby desired end-effector pose (blue ball for reference only) with high weight cost on left arm joint acceleration.

6.5 Discussion

This chapter presented a framework for optimal whole-body planning and control for dynamically stable mobile robots with multiple arms. The framework leverages the combination of simple centroidal dynamics and a full kinematics model instead of using a full-nonlinear dynamic model. The planning problem first solves a trajectory optimization problem offline. Then the same NLP is solved with a shorter time horizon in an MPC context to execute the motion. We define balancing for a ballbot in terms of the centroidal momentum instead of other approaches like ZMP or angular velocity that are more commonly used. The employed combination of the centroidal dynamics and a full kinematics model represented the system's dynamics sufficiently well to generate stable motion plans for the ballbot. We demonstrate the effectiveness of this algorithm performing different locomotion and arms motion tasks that require simultaneous control of the ball, body, and arms both in simulation and hardware. This framework has been shown in a ballbot with arms. Still, because of its high non-linearity, non-minimum phase behavior, and inherent instability, we believe it can be extended to other similar systems such as humanoids with wheeled feet and two-wheeled balancing manipulators.

The simulation results are considerably better than the results obtained in the hardware. As with any model-based planning and control algorithm, its performance is highly dependent on the robot model. To improve the implementation on the CMU ballbot, a more thorough system identification is required. Additionally, in our model, we do not consider the closed-loop dynamics of the ballbot with its balancing controller. Similar to [82], taking into account the inner closed-loop dynamics could improve the tracking performance. Moreover, in our current formulation, we do not consider the problem of force and contact planning. This can be realized by projecting the total wrench generated by external contacts to the centroidal dynamics.

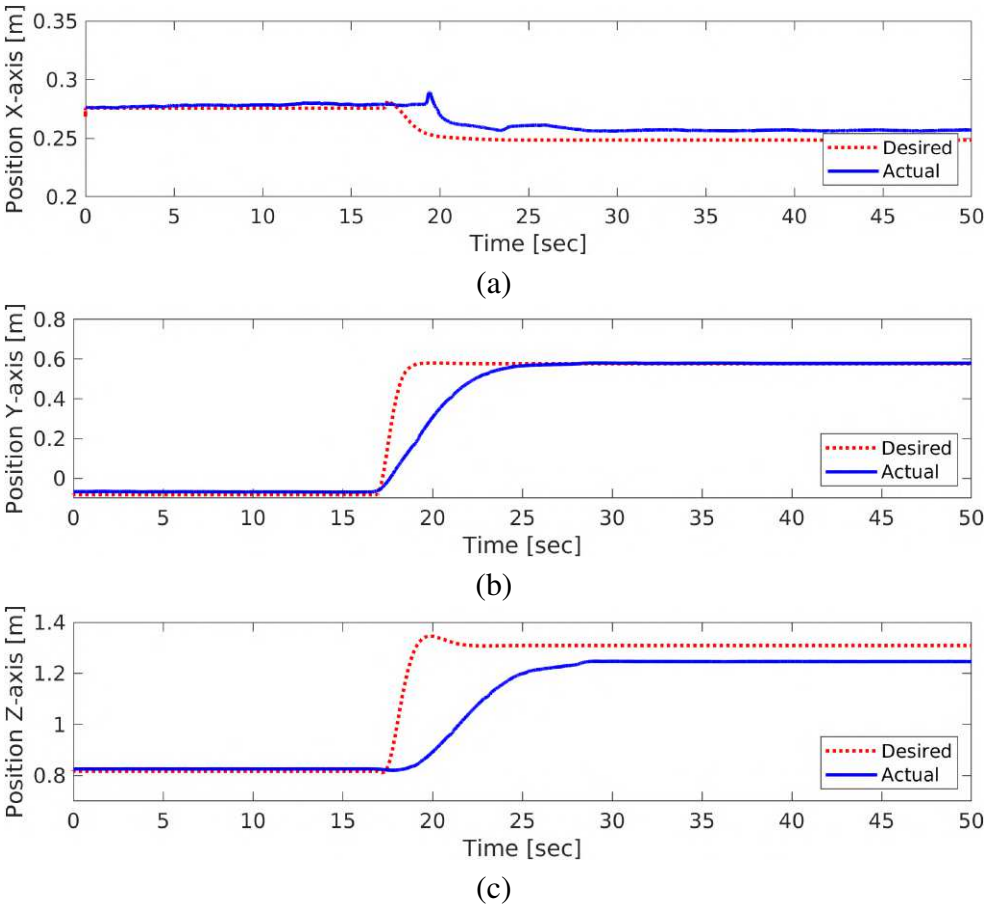
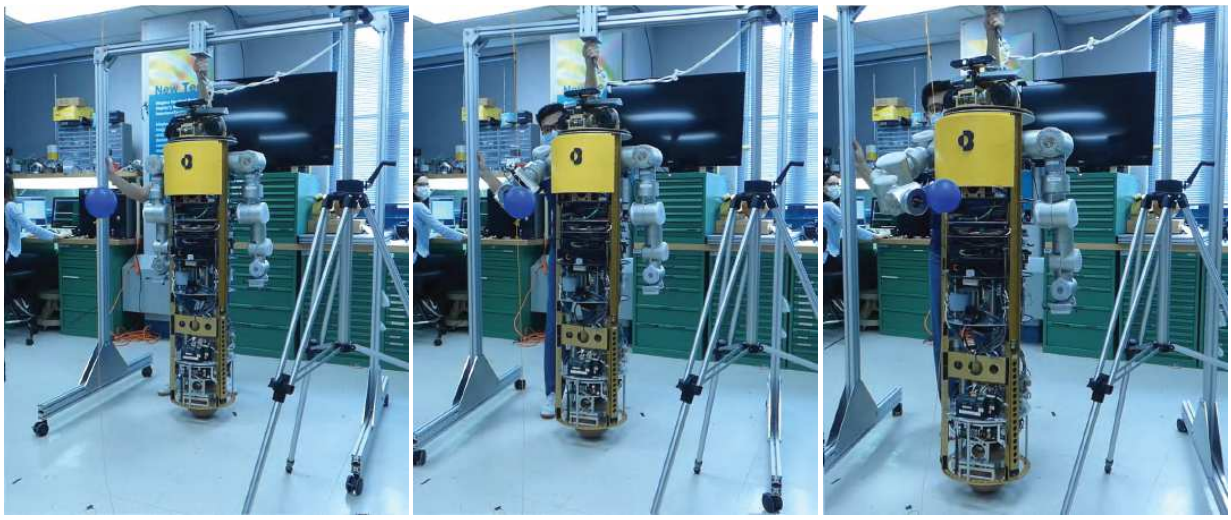


Figure 6.13: Experiment 1: The decomposed right end-effector position tracking performance of the whole-body controller. The translation trajectories in the (a) X-axis (left/right), (b) Y-axis (front/back) and (c) Z-axis (up/down).



(a) t = 0 sec (b) t = 3 sec (c) t = 5 sec

Figure 6.14: Screenshots of tracking a far away desired end-effector pose (blue ball for reference only) with high weight cost on left arm joint acceleration.

Chapter 7

Conclusion and Future Work

The ballbot's form factor and balancing capability have already been shown to be a very functional mobile robot capable of useful and interesting physical interactions. However, the ballbot as a mobile manipulation platform has been little studied before the work presented in this thesis. This thesis presents a first-of-its-kind mobile manipulation platform based on the human-size CMU ballbot with the addition of a pair of dexterous custom 7-DOF arms and hands.

The body of the work develops planning and control frameworks for the enhanced CMU ballbot to operate over a wide envelope of possible configurations. Two control philosophies are presented in this thesis: (i) a decoupled lower and upper body control strategy where the existing balancing controller compensates for the arm movement while the arms react to the body motion; and (ii) an optimal whole-body planning and control strategy that considers the entire kinematics and dynamics of the system in a single formulation.

This chapter highlights the contributions of the work presented in this thesis and discusses several possible future directions of the ballbot project.

7.1 Contributions

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

Demonstrating heavy box lifting and transportation with the ballbot

Chapter 3 presented a pragmatic center of mass compensation control strategy for shape-accelerating mobile manipulators, such as the ballbot. The controller enabled the ballbot with a pair of 2-DOF arms to lift and transport heavy payloads while retaining the ability to balance gracefully. The ballbot was able to carry a maximum load of 15 kg. This work proved that building a ballbot with more complex and higher inertia arms is possible. It also set the foundation for the development of controllers for the ballbot with 7-DOF arms based on COM compensation.

The design and development of pair of multi-DOF arms and hands

Chapter 4 presented the design, development, and integration of a 14-DOF dual manipulation system for the CMU ballbot. The system includes a pair of self-contained 7-DOF arms. Each arm weighs 12.9 kg, with a reach of 0.815 m, and a maximum payload of 7.5 kg at full extension. The ballbot's arms have a larger payload-to-weight ratio than

commercially available cobot arms with similar or greater payload capacity. Design features include highly integrated sensor-actuator-control units, torque sensors on each joint, lightweight exoskeleton structure, and anthropomorphic kinematics. A torque limiter at the shoulder joint was incorporated to protect against large impact forces. We integrated multiple passive and active end-effectors to provide dexterity. As default, the system has a pair of 3-Finger Barrett Hands.

A new configuration for mobile manipulation

The integration of the new pair of 7-DOF arms and hands to the existing human-size CMU ballbot results in a new robot configuration for mobile manipulation. To the best of our knowledge, the CMU ballbot is the first-of-its-kind. It is the first ballbot with a pair of anthropomorphic arms. The resulting robot configuration combines the smooth omnidirectional motion of the ballbot base with the new capability to interact with objects and the environment through manipulation. The new robot platform was designed with the aim of supporting future research directions.

A control algorithm to balance with the high DOF and inertia arms

Chapter 4 also presented a COM compensation strategy to extend the existing balancing controller. The controller enables the ballbot to balance and station-keep while moving its 7-DOF arms. The new arms have considerable mass and inertia. Consequently, their movement heavily affects the robot's stability. With the proposed feedforward control strategy the ballbot is able to effectively track its position on the floor, *i.e.*, ± 0.15 m, while swinging the arms at moderate speeds, *i.e.*, ± 0.6 rad/s (34.4 deg/s).

Development of a task space impedance control for the ballbot

For manipulation, we are interested in controlling the robot's end-effector. Additionally, it is desirable to have compliant arm control strategies that complement the compliance of the ballbot base. Chapter 5 extends the framework presented in section 4.5 to realize task space impedance control with the ballbot with a pair of 7-DOF arms. In a close-loop, the task space controller can react to the control inputs of the balancing controller.

Performance evaluation of the ballbot performing simple manipulation tasks

To evaluate the performance of the task space controller, several experiments were performed. It was shown that effective end-effector trajectory tracking and regulation were possible. Additionally, demonstrations of the controller realizing real manipulation tasks are shown.

A framework for whole-body optimal planning and control

Chapter 6 presented a framework for whole-body optimal planning and control for dynamically stable mobile robots with multiple arms. The framework presented implements centroidal dynamics and a full kinematics model instead of using a full-nonlinear dynamic model to describe the behavior of the robot. The control framework was implemented and tested on the ballbot. Using this method, the ballbot performed simple combined locomotion and manipulation tasks.

This thesis has also yielded several contributions regarding both software and project infrastructure. These contributions benefit the project as a whole and enable the ballbot to be used as a research platform by future students:

Hardware and software updates for the ballbot

The high-level Linux computer was upgraded to provide additional processing power and modern I/O ports as part of this thesis work. The new computer added USB 3.0 ports and a graphics card to connect two Intel RealSense cameras and do all the processing onboard. The robot's network interface was also upgraded to handle the traffic of the new Ethernet-controlled arms. The OS was upgraded from Ubuntu 14.04 to Ubuntu 16.04. The entire ROS software stack was upgraded from ROS Indigo to ROS Kinetic alongside the OS upgrade. This encompassed upgrading numerous ROS packages developed since 2012 by the research group. This enabled the use of the latest open-source software available in the ROS ecosystem.

Arm's low-level control and ROS integration

In addition to the hardware development of the arms, the low-level software stack for the arms was developed. A C++ library was developed to provide an easy interface to the arm's input commands and feedback signals. The arms were integrated with the ROS hardware interface. This enabled the support of the standard ROS interfaces, and taking advantage of a large library of open-source controllers for robot arms.

Physics simulation environment of the ballbot

As part of this thesis work, a new physics simulation environment of the enhanced ballbot was developed in PyBullet [96]. The new simulation environment provides a 1:1 ROS communication interface replica to the robot's real-time QNX computer. The simulation provides a fast and easy sandbox to test new planning and control algorithms in a diverse set of environments.

Ballbot Wiki

A significant amount of effort has been put into the documentation to enable this new research platform to be easily used by future students and researchers. We created a wiki page that contains general information and parameters of the ballbot, maintenance instructions, operation instructions, common issues and how to debug them, and code development rules. A new user of the platform should be able to run the ballbot and develop new code for it by reading through the documentation.

This thesis furthers state of the art in balancing mobile manipulator robots. The principal merit lies, in part, in proving the hypothesis that the resulting first-of-its-kind, highly dynamic multi-DOF configuration is controllable over a wide envelope of possible configurations. By developing hardware, control algorithms, and testing experimentally on the ballbot, this work has demonstrated basic capabilities that includes; accurately moving the arms/hands to points in space while dynamically balancing; adapting to external disturbances and varying payloads; and grasping objects fixed in the environment.

7.2 Future Work

The ballbot is unlike any mobile robot that came before it. It is the very first mobile robot to balance on a spherical wheel. This innovation yields the benefits of physical compliance and an advantageous form factor. This thesis adds manipulation capabilities to the ballbot and

explores some simple manipulation tasks. These new capabilities enable a vast range of future possibilities for the platform. This section presents brief descriptions of different improvements and extensions that can be applied to the work presented in this thesis.

7.2.1 Robust Force Planning and Control

The task space impedance control framework presented in Chapter 5 provides a hybrid position/force interface for the ballbot arms. However, this is just a reactive base controller. For a robot to be able to truly interact with its environment it has to be able to plan, apply and handle external forces. It may also need to use its body dynamics to apply a greater force.

The whole-body momentum based planning framework presented in Chapter 6 can be easily extended to plan for end-effector forces and torques. The rate of centroidal linear and angular momentum $\dot{h}(q, \dot{q}) = [\dot{k}, \dot{l}]^T$, computed from the robot's joint angles and velocities, equals the total wrench generated by the external contacts and the gravitational forces:

$$\dot{k} = m\ddot{r} = \sum_j F_j + mg \tag{7.1}$$

$$\dot{l}(q, \dot{q}) = \sum_j (c_j - r) \times F_j + \tau_j \tag{7.2}$$

where m is the total mass of the robot, $r \in \mathbb{R}^3$ is the COM position, $F_j \in \mathbb{R}^3$ is the external contact force at j^{th} contact point, $c_j \in \mathbb{R}^3$ is the position of the j^{th} contact point, $\tau_j \in \mathbb{R}^3$ is the torque at the j^{th} contact point, and $g \in \mathbb{R}^3$ is the gravitational acceleration.

Leveraging this information we can add the set of F_j and τ_j for all contact points as decision variables to the optimization problem. This will allow the optimization to solve not only over joint space but also over the external wrench space. This fundamental capability to plan and control in the force space will be useful for dynamic tasks such as pushing off walls and pushing heavy objects.

7.2.2 Improve loco-manipulation planning and control

In Chapter 6 we presented a framework for whole-body planning and control. The implemented framework on the ballbot demonstrated the successful execution of combined locomotion and manipulation tasks simultaneously. However, the control architecture could be improved to remove the undesirable oscillations of the robot ball position. At the lowest level of the control architecture is the balancing controller. This controller introduces delays into the dynamics of the systems that are not considered by the planning and higher-level tracking controller. Incorporating information of the balancing controller into the model-based planners and controllers could potentially improve the overall performance. Other control strategies that do not include the low-level balancing controller could also be investigated. Having direct torque control over the ball actuators could improve tracking of the whole-body motion plan.

7.2.3 Dynamic Interactions for Mobile Manipulation

There has been a great deal of recent interest in robots working with or alongside people in critical applications, from manufacturing to health care and quality of life. In many cases, the interaction involves direct physical contact between robot and human. Despite exciting research progress, there remains a need for robots that can fully utilize the dynamics of their bodies, along with affordances in the environment, to accomplish challenging tasks in an efficient, robust, and safe manner. Many common interactions utilize significant dynamics. For example, when opening a heavy door or initiating the motion of a wheelchair, people use the dynamic motion of their bodies to make the task easier. Handholds may be used to stabilize a dynamic interaction or to facilitate rapid movement through cluttered spaces. Few existing robots are capable of such abilities, especially while working with people. The ballbot's unique dynamics make it an exciting platform to investigate dynamic interactions.

To realize such dynamic physical interactions it is required to explore new interaction models and methods for planning, learning, and real-time control. Observations of human behavior show that dynamic whole-body interactions that exploit upper body contact with the environment are common to realize complex dynamic tasks. Human subjects studies can be performed to develop dynamic interaction models and explore the ideas of reflex models for short-term contact events and dynamic skill models for longer-term coordinated interactions.

Assuming that humans are experts at performing dynamic interaction tasks, the results of the human studies should be transferred to the robot, potentially using imitation learning. For dynamic tasks, simply repeating the demonstrated trajectories on the ballbot will not suffice as the robot and the human have very different kinematics and dynamics [97, 98, 99]. Thus, there is a need to develop analytical and learning-based methods for mapping human demonstrations onto the robot's skills.

7.2.4 Maneuvering a Wheelchair

The CMU ballbot has evolved into a competent and exciting robot that has particular relevance to the field of physical human-robot interaction. Studies with the ballbot platform focused on possible application scenarios where the technology may assist the elderly or physically handicapped can advance whole-body dynamic interaction control limits. An integrative task demonstrating various skills working together is maneuvering a wheelchair.

To perform the task, the ballbot must position itself relative to the wheelchair and identify and grasp the two wheelchair handles using vision subtasks that are difficult in themselves. The ballbot must then accurately control its arms to apply the correct forces to go in the desired direction using its navigation ability while always respecting the wheelchair's dynamics and kinematics. It will need to apply enough force through its body lean or arms to overcome high friction and go over bumps and ramps. The control algorithm could use continuous online learning to robustly handle unknown disturbances such as passenger mass and wheelchair friction characteristics that may cause the chair to drift to one side or require extra forces to roll the wheelchair. Leveraging the ballbot's unique dynamic balancing and lean strength capabilities, it would be possible to demonstrate navigating an ADA-compliant 4.8° slope with a 114 kg (250 lb.) load. This would require forces of approximately 96 N that are within the ballbot's capability.

Control experiments and human studies should be conducted to evaluate the effectiveness (*i.e.*, how accurate and fast can the ballbot reach its destination), reliability (*i.e.*, how repeatable can the task be performed successfully), and comfort (*i.e.*, how comfortable a human subject feels on the wheelchair) performing the complete task.

7.2.5 Accurate Payload Estimation and Adaptation

As described throughout the thesis for balancing mobile manipulator robots, like the ballbot, locomotion and manipulations tasks are tightly coupled. This brings about some challenges. One of the challenges is that the manipulated object's weight plays a significant role in the robot's balance and locomotion. We introduce a simple method to estimate the payload's weight from the joint torques and compensate for its dynamic effect. However, there are significant errors in the estimate. A better state estimator that actively estimates the object mass will improve performance and ease performing manipulation tasks. Further, the whole-body control framework in Chapter 6 should be improved to account for the object's physical properties and dynamics to realize smooth trajectories.

7.2.6 Dynamic Cooperative Carrying

The new pair of 7-DOF arms and hands presented in Chapter 4 are more powerful, enabling new opportunities to study tasks that require carrying, pushing, and pulling heavier objects.

A task of potential interest is collaborative carrying. Strenuous load transportation tasks performed solely by humans have the potential of causing accidents and repetitive stress injuries. Dynamic, collaborative load manipulation skills between a robotic agent and a human will enable an alternative, reliable, safe transportation and manipulation framework. The ballbot's inherit omnidirectional compliance, and strong arms make it an ideal platform to study this task. The ballbot and human must share the object's load since it is too large and cumbersome to be carried alone. The ballbot can bring an object close to its body and lean backward to carry part of the load with its body, similar to how humans do. Given the importance of human-robot coordination, three cases can be investigated: (i) the human leads the robot, (ii) the robot leads the human, and (iii) the most interesting case where the human and robot have a shared plan for how to carry the object through an environment [100].

Appendix A

SENSO-Joints' Sensor Specification

Table A.1: SENSO-Joints' Sensor Specification

	SENSO-Joint 100 RD5014	**SENSO-Joint 100 RD5008**	**SENSO-Joint 75 RD3806**
Incremental Encoder	RLS RLC2IC		
Resolution	13 bit		
Counts per Revolution	4096		3200
Accuracy	± 0.40		
Magnet	RLS MR040G		RLS MR031G
Absolute Encoder	RLS Orbis		RD50-AksIM
Resolution	14 bit		16 bit
Counts per Revolution	16384		65536
Accuracy	$\pm 0.25^\circ$		$\pm 0.1^\circ$
Magnet	Orbis magnet		MRA2
Torque Sensor			
Nominal Torque	67 Nm	40 Nm	10 Nm
Maximum Torque	200 Nm	80 Nm	20 Nm
Accuracy	± 0.6 Nm	± 0.4 Nm	± 0.1 Nm

Appendix B

Task Space Dynamics

This section presents the derivation of the ballbot dynamics in task space for the development of a task space impedance control law in Chapter 5. The joint space dynamic model of the ballbot's arm with respect to the generalized joint coordinates $\boldsymbol{q} \in \mathbb{R}^n$ can be expressed as:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\epsilon} = \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}^T(q)\boldsymbol{F}_{ext} \tag{B.1}$$

However, the desired dynamics are defined in task spaced coordinates $\boldsymbol{x} \in \mathbb{R}^m$. It is necessary to transform the model in (B.1) from joint coordinates to task space coordinates.

The relationship between the Cartesian coordinates $\boldsymbol{x}$ and the joint configuration coordinates $\boldsymbol{q}$ is given by the forward kinematics function $\mathbf{FK} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, i.e. $\boldsymbol{x} = \mathbf{FK}(\boldsymbol{q})$. In our formulation the number of joint configuration of the arm is $n = 7$ and the number of Cartesian coordinates is $m = 6$. Thus, our system is redundant because $n > m$. Let

$$J(q_q) = \frac{\delta FK(q_a)}{\delta q_a} \in \mathbb{R}^{n \times m} \tag{B.2}$$

be the Jacobian matrix that relates the joint velocities $\dot{q} \in \mathbb{R}^n$ to the task velocity $\dot{x} \in \mathbb{R}^m$ according to

$$\dot{x} = J(q)\dot{q}. \tag{B.3}$$

Taking the derivative of (5.5) the relation between joint acceleration $\ddot{q}_a \in \mathbb{R}^n$ to the task acceleration $\ddot{x} \in \mathbb{R}^m$ is obtained as:

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q)\dot{q}. \tag{B.4}$$

The Jacobian matrix $J(q)$ also relates the external torque vector τ_{ext} to the generalized external forces F_{ext} by

$$\tau_{ext} = J(q)^T F_{ext}. \tag{B.5}$$

To project the joint space dynamics to the task space dynamics we first pre-multiply (B.1) by JM^{-1}, we get

$$J\ddot{q}_a + JM^{-1}(C\dot{q}_a + g + \tau_{fric}) = JM^{-1}\tau + JM^{-1}\tau_{ext}. \tag{B.6}$$

Substituting equation (B.3), (B.4), (B.5) into the resulting (B.6), we get

$$\ddot{x} - \dot{J}\dot{q} + JM^{-1}h = JM^{-1}\tau + JM^{-1}J^T F_{ext}, \tag{B.7}$$

where $h = C\dot{q} + g + t_{fric}$. The joint space inertia matrix M can be transformed into its equivalent task space inertia Λ as:

$$\Lambda = (JM^{-1}J^T)^{-1}. \tag{B.8}$$

Assuming that the joint torques τ and a virtual force F_τ at the EE are related by $\tau = J^T F_\tau$ and pre-multiplying (B.7) by task space inertia Λ we get the task space dynamics:

$$\Lambda\ddot{x} + \Lambda(JM^{-1}h - \dot{J}\dot{q}) = \Lambda JM^{-1}J^T F_\tau + F_{ext}. \tag{B.9}$$

Since $\Lambda JM^{-1}J^T = I$ we can simply the equation to

$$\Lambda\ddot{x} + \mu = F_\tau + F_{ext}, \tag{B.10}$$

where $\mu = \Lambda(JM^{-1}h - \dot{J}\dot{q})$. The objective of the control law is to find a $F_\tau = F_\tau^*$ such that the real dynamics in (5.9) follow some desired dynamic behaviour. For *Task Space Impedance Control* a popular choice of desired dynamics are those of a *mass-spring-damper system* of the form:

$$\bar{M}_d\ddot{e}_x + \bar{B}_d\dot{e}_x + \bar{K}_d e_x = e_f, \tag{B.11}$$

where $\bar{M}_d$, $\bar{B}_d$, and $\bar{K}_d$ are the desired inertia, damping and stiffness of the virtual system. $e_x, \dot{e}_x$, and $\ddot{e}_x$ are the end-effector pose errors between the actual and desired pose. e_f is the error between the desired applied force and the actual force. These errors can be calculated using different methods depending on the representation of the task coordinates.

One approach to computing F_τ^* is to compute the task acceleration $\ddot{x}$ from (B.11) as:

$$\ddot{x} = \bar{M}_d^{-1}(F_d - F_{ext} - \bar{B}_d\dot{e}_x - \bar{K}_d e_x) + \ddot{x}_d \tag{B.12}$$

and plug-in back into (B.10) to get

$$\Lambda(\bar{M}_d^{-1}(F_d - F_{ext} - \bar{B}_d\dot{e}_x - \bar{K}_d e_x) + \ddot{x}_d) + \mu = F_\tau^* + F_{ext}, \tag{B.13}$$

and solving for the desired virtual force to get

$$F_\tau^* = \Lambda\ddot{x}_d - \Lambda\bar{M}_d^{-1}(\bar{B}_d\dot{e}_x + \bar{K}_d e_x) + \mu + \Lambda\bar{M}_d^{-1}F_d - (I + \Lambda\bar{M}_d^{-1})F_{ext}. \tag{B.14}$$

From the assumption that $\tau = J^T F_\tau$ we get our control law in joint space to be:

$$\tau^* = J^T\left[\Lambda\ddot{x}_d - \Lambda\bar{M}_d^{-1}(\bar{B}_d\dot{e}_x + \bar{K}_d e_x) + \mu + \Lambda\bar{M}_d^{-1}e_F - F_{ext}\right]. \tag{B.15}$$

The total joint torque control law (5.30) can be split into its feedforward τ_{ff} and feedback τ_{fb} components as,

$$\tau^* = \tau_{ff} + \tau_{fb}, \tag{B.16}$$

where

$$\tau_{ff} = J^T\left[\Lambda\ddot{x}_d - \mu - F_{ext}\right] \tag{B.17}$$

and

$$\tau_{fb} = -J^T\left[\Lambda\bar{M}_d^{-1}(\bar{B}_d^{-1}\dot{e}_x + \bar{K}_d e_x + e_f)\right]. \tag{B.18}$$

Appendix C

Links to the ballbot videos

Chapters 3- 6 presented several successful experimental results on the ballbot, and links to the videos of the ballbot achieving these results are listed below.

C.1 Lifting heavy payloads with 2-DOF arms

https://youtu.be/lVUEqauRmII

C.2 Initial experiments of the ballbot with 7-DOF arms

https://youtu.be/FYKJiXFJrlE

C.3 Arm choreography while station-keeping

https://youtu.be/D33xcA4cqd0

C.4 Ballbot lifting a heavy payload with 7-DOF arms

https://youtu.be/P1KXYxAo2-k

C.5 Task space control results

End-effector pose regulation and balancing a “wine glass”.
https://youtu.be/lBq_djcovVQ

C.6 End-effector circular motion

https://www.youtube.com/watch?v=6DmeSe4HqPY

C.7 Whole-Body Motion Planning Simulation Results

https://youtu.be/EDU-12m-4gg

Bibliography

[1] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the PR2," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5568–5575.

[2] M. Fuchs, C. Borst, P. R. Giordano, A. Baumann, E. Kraemer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze *et al.*, "Rollin'Justin-Design considerations and realization of a mobile platform for a humanoid upper body," in *Robotics and Automation, (ICRA) International Conference on*. IEEE, 2009, pp. 4131–4137.

[3] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on autonomous mobile service robots*, 2016.

[4] M. Stilman, J. Olson, and W. Gloss, "Golem Krang: Dynamically stable humanoid robot for mobile manipulation," in *Robotics and Automation, (ICRA) International Conference on*. IEEE, 2010, pp. 3304–3309.

[5] Boston Dynamics, "Handle". [Online]. Available: https://www.bostondynamics.com/handle

[6] Boston Dynamics, "Atlas". [Online]. Available: https://www.bostondynamics.com/atlas

[7] M. Shomin, J. Forlizzi, and R. Hollis, "Sit-to-Stand Assistance with a Balancing Mobile Robot," in *IEEE Int'l. Conf. on Robotics and Automation*, Seattle, WA, May 26-30 2015.

[8] M. Shomin, "Navigation and Physical Interaction with Balancing Robots," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, October 2016.

[9] X. Huo, Y. Liu, L. Jiang, and H. Liu, "Design and development of a 7-DOF humanoid arm," in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*. IEEE, 2012, pp. 277–282.

[10] D. Bauer, C. Bauer, A. Lakshmipathy, R. Shu, and N. S. Pollard, "Towards Very Low-Cost Iterative Prototyping for Fully Printable Dexterous Soft Robotic Hands," [Manuscript submitted for publication]. The Robotics Institute, Carnegie Mellon University.

[11] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-Body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[12] T. Lauwers, G. Kantor, and R. Hollis, "One is Enough!" in *Proc. Int'l. Symp. for Robotics*

Research. San Francisco: Int'l. Foundation for Robotics Research, October 12-15 2005.

[13] U. Nagarajan, G. Kantor, and R. Hollis, "The Ballbot: An Omnidirectional Balancing Mobile Robot," *International Journal of Robotics Research*, vol. 33, no. 6, pp. 917–930, May 2014.

[14] U. Nagarajan, G. Kantor, and R. L. Hollis, "Human-robot physical interaction with dynamically stable mobile robots," in *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, 2009, pp. 281–282.

[15] U. Nagarajan, B. Kim, and R. Hollis, "Planning in high-dimensional shape space for a single-wheeled balancing mobile robot with arms," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 130–135.

[16] U. Nagarajan, G. Kantor, and R. Hollis, "Hybrid Control for Navigation of Shape-Accelerated Underactuated Balancing Systems," in *IEEE Conf. on Decision and Control*, Atlanta, GA, December 15-17 2010.

[17] ——, "Integrated Planning and Control for Graceful Navigation of Shape-Accelerated Underactuated Balancing Mobile Robots," in *International Journal of Robotics Research*, vol. 32, no. 9-10, September 2013, pp. 1005–1029.

[18] M. Kumagai and T. Ochiai, "Development of a robot balancing on a ball," in *2008 International Conference on Control, Automation and Systems*. IEEE, 2008, pp. 433–438.

[19] ——, "Development of a Robot Balanced on a Ball–First Report, Implementation of the Robot and Basic Control–," *Journal of Robotics and Mechatronics*, vol. 22, no. 3, pp. 348–355, 2010.

[20] P. Fankhauser and C. Gwerder, "Modeling and control of a ballbot," B.S. thesis, Eidgenössische Technische Hochschule Zürich, 2010.

[21] S. B. Sankhyan and G. K. Wadhwa, "Ball Balancing Robot: Future of Transportation," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2018, pp. 299–301.

[22] T. K. Jespersen, "Kugle-Modelling and Control of a Ball-balancing robot," *Master Thesis, Aalborg University, Aalborg, Denmark*, 2019.

[23] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-Body MPC for a dynamically stable mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3687–3694, 2019.

[24] U. Nagarajan, G. Kantor, and R. Hollis, "Integrated planning and control for graceful navigation of shape-accelerated underactuated balancing mobile robots," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 136–141.

[25] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground," *Autonomous Robots*, vol. 33, no. 4, pp. 399–414, 2012.

[26] R. Hollis, "Ballbots," *Scientific American*, vol. 295, no. 4, pp. 72–77, 2006.

[27] U. Nagarajan, G. Kantor, and R. Hollis, "Trajectory Planning and Control of an Underac-

tuated Dynamically Stable Single Spherical Wheeled Mobile Robot," in *Proc. IEEE Int'l. Conf. on Robotics and Automation*, May 2009.

[28] T. B. Lauwers, G. A. Kantor, and R. L. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2884–2889.

[29] A. K. Mampetta, "Automatic transition of ballbot from statically stable state to dynamically stable state," *Report CMU-RI-TR-01–00*, 2006.

[30] U. Nagarajan, A. Mampetta, G. A. Kantor, and R. L. Hollis, "State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot," in *Proc. IEEE Int'l. Conf. on Robotics and Automation*, Kobe, Japan, May 12-17 2009.

[31] E. M. Schearer, "Modeling dynamics and exploring control of a single-wheeled dynamically stable mobile robot with arms," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., 2006.

[32] U. Nagarajan, "Dynamic Constraint-based Optimal Shape Trajectory Planner for Shape-Accelerated Underactuated Balancing Systems," in *Robotics: Science and Systems*, Zaragoza, Spain, June 27-31 2010.

[33] M. Shomin and R. Hollis, "Differentially flat trajectory generation for a dynamically stable mobile robot," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4467–4472.

[34] ——, "Fast, Dynamic Trajectory Planning for a Dynamically Stable Mobile Robot," in *IEEE/RSJ Int'l. Symposium on Robotics and Systems (IROS)*, September 2014.

[35] B. Vaidya, M. Shomin, R. Hollis, and G. Kantor, "Operation of the ballbot on slopes and with center-of-mass offsets," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2383–2388.

[36] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[37] C. Crick, G. Jay, S. Osentoski, and O. C. Jenkins, "ROS and rosbridge: Roboticists out of the loop," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2012, pp. 493–494.

[38] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ROS_control: A generic and simple control framework for ROS," *The Journal of Open Source Software*, 2017. [Online]. Available: http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf

[39] P. Asgari, P. Zarafshan, and S. A. A. Moosavian, "Dynamics modelling and stable motion control of a ballbot equipped with a manipulator," in *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. IEEE, 2013, pp. 259–264.

[40] U. Nagarajan and R. Hollis, "Shape space planner for shape-accelerated balancing mobile robots," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1323–1341, 2013.

[41] F. Sonnleitner, R. Shu, and R. L. Hollis, "The Mechanics and Control of Leaning to Lift Heavy Objects with a Dynamically Stable Mobile Robot," *Int'l. Conf. on Robotics and Automation*, May 20-24 2019.

[42] M. W. Spong, "Underactuated Mechanical Systems," in *Control problems in robotics and automation*. Springer, 1998, pp. 135–150.

[43] U. Nagarajan, "Fast and graceful balancing mobile robots," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, July 2012.

[44] Z. Li and R. Hollis, "Toward A Ballbot for Physically Leading People: A Human-Centered Approach," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4827–4833.

[45] "ABB's Collaborative Robot -YuMi - Industrial Robots From ABB Robotics." [Online]. Available: https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi

[46] "Yaskawa MOTOMAN SDA10D." [Online]. Available: https://www.motoman.com/en-us/products/robots/industrial/assembly-handling/sda-series/sda10d

[47] "KAWADA NEXTAGE: a next-generation robot able to work together with people." [Online]. Available: http://nextage.kawada.jp/en/

[48] G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl, "DLR's torque-controlled light weight robot III-are we reaching the technological limits now?" in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2. IEEE, 2002, pp. 1710–1716.

[49] Universal Robots, "UR10". [Online]. Available: https://www.universal-robots.com/products/ur10-robot/

[50] F. E. GmbH, "Franka Emika Panda." [Online]. Available: https://www.franka.de/technology

[51] Kinova Robotics, "Gen 3 Robot". [Online]. Available: https://www.kinovarobotics.com/en/products/professional-robots

[52] S. Rader, L. Kaul, H. Fischbach, N. Vahrenkamp, and T. Asfour, "Design of a high-performance humanoid dual arm system with inner shoulder joints," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 523–529.

[53] L. Baccelliere, N. Kashiri, L. Muratore, A. Laurenzi, M. Kamedula, A. Margan, S. Cordasco, J. Malzahn, and N. G. Tsagarakis, "Development of a human size and strength compliant bi-manual platform for realistic heavy manipulation tasks," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 5594–5601.

[54] D. Russo, T. Foley, K. Stroud, J. Connolly, B. Tillman, and L. Pickett, "NASA space flight human system standards," in *Proceedings of the Human Factors and Ergonomics Society*

Annual Meeting, vol. 51, no. 21. SAGE Publications Sage CA: Los Angeles, CA, 2007, pp. 1468–1470.

[55] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the DLR-III lightweight manipulator arm," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 1623–1630.

[56] NASA, "Man-Systems Integration Standards (MSIS) - NASA-STD-3000." [Online]. Available: https://msis.jsc.nasa.gov/sections/section03.htm

[57] KUKA, "KUKA LBR iiwa." [Online]. Available: https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa

[58] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*. Springer, 2017, vol. 118.

[59] Barrett Technology, "Barrett Hand". [Online]. Available: https://advanced.barrett.com/barretthand

[60] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Resolved-acceleration control of robot manipulators: A critical review with experiments," *Robotica*, vol. 16, no. 5, pp. 565–573, 1998.

[61] S.-D. Lee, Y.-L. Kim, and J.-B. Song, "Novel collision detection index based on joint torque sensors for a redundant manipulator," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4636–4641.

[62] N. Hogan, "Impedance control: An approach to manipulation: Part i—theory," *Journal of Dynamic Systems, Measurement, and Control*, 1985.

[63] ——, "Impedance control: An approach to manipulation: Part ii—implementation," *Journal of Dynamic Systems, Measurement, and Control*, 1985.

[64] C. Ott, *Cartesian impedance control of redundant and flexible-joint robots*. Springer, 2008.

[65] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

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

[67] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.

[68] M. Mistry and L. Righetti, "Operational space control of constrained and underactuated systems," in *Robotics: Science and systems*, vol. 7, 2012, pp. 225–232.

[69] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter, "Alma-articulated locomotion and manipulation for a torque-controllable robot," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8477–8483.

[70] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body

control for the atlas robot," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 120–127.

[71] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, "Quadratic programming for multirobot and task-space force control," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 64–77, 2019.

[72] A. Heins, M. Jakob, and A. P. Schoellig, "Mobile manipulation in unknown environments with differential inverse kinematics control," in *2021 18th Conference on Robots and Vision (CRV)*. IEEE, 2021, pp. 64–71.

[73] L. Sentis and O. Khatib, "A Whole-Body control framework for humanoids operating in human environments," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2641–2648.

[74] ——, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 505–518, 2005.

[75] A. Dietrich, K. Bussmann, F. Petit, P. Kotyczka, C. Ott, B. Lohmann, and A. Albu-Schäffer, "Whole-body impedance control of wheeled mobile manipulators," *Autonomous Robots*, vol. 40, no. 3, pp. 505–517, 2016.

[76] M. Shomin and R. Hollis, "Fast, dynamic trajectory planning for a dynamically stable mobile robot," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 3636–3641.

[77] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0031320314000235

[78] R. Shu and R. Hollis, "Momentum based whole-body optimal planning for a single-spherical-wheeled balancing mobile manipulator," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3221–3226.

[79] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for Whole-Body control," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1550039, 2016.

[80] A. Rocchi, E. M. Hoffman, D. G. Caldwell, and N. G. Tsagarakis, "OpenSoT: A Whole-Body control library for the compliant humanoid robot COMAN," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6248–6253.

[81] B. Henze, A. Dietrich, and C. Ott, "An approach to combine balancing with hierarchical Whole-Body control for legged humanoid robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 700–707, 2015.

[82] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control," *arXiv preprint arXiv:1909.06586*, 2019.

[83] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for

quadrupedal locomotion: Coupled planning, terrain mapping, and Whole-Body control," *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1635–1648, 2020.

[84] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[85] S. Xin and S. Vijayakumar, "Online Dynamic Motion Planning and Control for Wheeled Biped Robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 3892–3899.

[86] A. Patel, S. L. Shield, S. Kazi, A. M. Johnson, and L. T. Biegler, "Contact-implicit trajectory optimization using orthogonal collocation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2242–2249, 2019.

[87] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[88] S. Xin, Y. You, C. Zhou, C. Fang, and N. Tsagarakis, "A torque-controlled humanoid robot riding on a two-wheeled mobile platform," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1435–1442.

[89] S. Dafarra, S. Bertrand, R. J. Griffin, G. Metta, D. Pucci, and J. Pratt, "Non-Linear Trajectory Optimization for Large Step-Ups: Application to the Humanoid Robot Atlas," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 3884–3891.

[90] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2589–2594.

[91] J. Betts and I. Kolmanovsky, "Practical methods for optimal control using nonlinear programming," *Applied Mechanics Reviews*, vol. 55, p. B68, 2002.

[92] R. Shu, A. Siravuru, A. Rai, T. Dear, K. Sreenath, and H. Choset, "Optimal control for geometric motion planning of a robot diver," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4780–4785.

[93] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[94] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[95] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[96] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[97] K. Muelling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research (IJRR)*, no. 3, pp. 263–279, 2013.

[98] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with EM-based reinforcement learning," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, October 2010, pp. 3232–3237.

[99] T. Osa, J. Pajarinen, G. Neumann, J. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, pp. 1–179, Mar. 2018.

[100] A. Mörtl, M. Lawitzky, A. Kucukyilmaz, M. Sezgin, C. Basdogan, and S. Hirche, "The role of roles: Physical cooperation between humans and robots," *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1656–1674, 2012.