LOCOMOTION AND BALANCE CONTROL OF HUMANOID ROBOTS WITH DYNAMIC AND KINEMATIC CONSTRAINTS

Yu Zheng

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2014

Approved by:

Katsu Yamane

Ming C. Lin

Dinesh Manocha

Ron Alterovitz

Jack Snoeyink

ABSTRACT

Yu Zheng: Locomotion and Balance Control of Humanoid Robots
with Dynamic and Kinematic Constraints
(Under the direction of Katsu Yamane and Ming C. Lin)

Building a robot capable of servicing and assisting people is one of the ultimate goals in humanoid robotics. To realize this goal, a humanoid robot first needs to be able to perform some fundamental locomotion tasks, such as balancing and walking. However, simply performing such basic tasks in static, open environments is insufficient for a robot to be useful. A humanoid robot should also possess the ability to make use of the object in the environment to generate dynamic motions and improve its mobility. Also, since humanoid robots are expected to work and live closely with humans, having human-like motions is important for them to be human-friendly.

This dissertation addresses my work on endowing humanoid robots with the ability to handle dynamic and kinematic constraints while performing the basic tasks in order to achieve more complex locomotion tasks. First, as a representative case of handling dynamic constraints, a biped humanoid robot is required to balance and walk on a cylinder that rolls freely on the ground. This task is difficult even for humans. I introduce a control method for a humanoid robot to execute this challenging task. In order for the robot to be able to actively control cylinder's motion, the dynamics of the cylinder has been taken into account together with the dynamics of the robot in deriving the control method. Its effectiveness has been verified by full-body dynamics simulation and hardware experiments on the Sarcos humanoid robot. Second, as an example of tasks with kinematic constraints, I present a method for real-time control of humanoid robots to track human motions while maintaining balance. It consists of a standard proportional-derivative tracking controller

that computes the desired acceleration to track the given reference motion and an optimizer that computes the optimal joint torques and contact forces to realize the desired acceleration, considering the full-body dynamics of the robot and strict constraints on contact forces. By taking advantage of the property that the joint torques do not contribute to the six degrees of freedom of the floating base, I decouple the computation of joint torques and contact forces such that the optimization problem with strict contact force constraints can be solved in real time. In full-body simulation, a humanoid robot is able to imitate various human motions by using this method.

Through this work, I demonstrate that considering dynamic and kinematic constraints in the environment in the design of controllers enables humanoid robots to achieve more complex locomotion tasks, such as manipulating a dynamic object or tracking given reference motions, while maintaining balance.

To my parents, Jian-Rong Zheng and Qiao-Di Sun, and my wife, Juan Du.

ACKNOWLEDGEMENTS

The past four and a half years I spent at the Department of Computer Science of the University of North Carolina (UNC) at Chapel Hill and Disney Research Pittsburgh (DRP) on the research that eventually turned into this dissertation have been truly memorable. I would like to thank people who made my journey so special and extraordinary. First of all, I would like to thank my advisor at UNC, Prof. Ming Lin, and my advisor at DRP, Dr. Katsu Yamane, for their elaborate guidance and altruistic support, as well as the freedom I was provided throughout my work. I would also like to thank my co-advisor at UNC, Prof. Dinesh Manocha, for his collaboration and invaluable suggestions. I would also like to thank the other members of my committee, Prof. Ron Alterovitz and Prof. Jack Snoeyink, for their insightful feedback and discussions on my research work and this dissertation.

I would like to thank many other talented collaborators, colleagues, and friends who have helped me with my work and study, including Matt Glisson, Michael Mistry, Akihiko Murai, Hengchin Yeh, Feng Zheng, and Tian Cao. I would also like to thank many anonymous reviewers who have helped me improve the quality of my work.

I want to express my special thanks to Disney Research Pittsburgh for having me there and supporting my work and also Carnegie Mellon University, especially Prof. Jessica Hodgins and Prof. Chris Atkeson, for letting me use the CMU Sarcos Humanoid Robot for my experiments.

Finally, I am deeply indebted to my parents, Jian-Rong Zheng and Qiao-Di Sun, and my wife, Juan Du, for their constant support and encouragement, without which this work would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES xi

LIST OF FIGURES xii

LIST OF ABBREVIATIONS xvi

1 INTRODUCTION 1

1.1 Locomotion Generation Overview 2

1.1.1 Environmental Property 3

1.1.2 Robot Model 3

1.1.3 Sensor Information 3

1.1.4 Framework for Motion Generation and Verification 4

1.2 Locomotion of Humanoid Robots 5

1.2.1 Robot Balance Control 5

1.2.2 Biped Locomotion Generation 6

1.2.3 Motion Tracking Control 7

1.3 Thesis Statement 8

1.4 Main Results 9

1.4.1 Balance Control in a Dynamic Environment 9

1.4.2 Manipulating a Dynamic Object by Active Walking 9

1.4.3 Motion Tracking under Strict Contact Constraints 10

2 BALANCE CONTROL ON A DYNAMIC OBJECT 11

2.1 Introduction 11

2.1.1 Main Results 11

2.1.2 Organization 12

2.2 Previous Work 12
2.3 Problem Overview 13
2.3.1 Problem Statement 13
2.3.2 Control Framework 14
2.4 Balance Controller 15
2.4.1 Flat-Foot Sagittal-Plane Simplified Model 16
2.4.2 Geta-Foot Sagittal-Plane Simplified Model 18
2.4.3 Frontal-Plane Simplified Model 18
2.4.4 Balance Controller for the Sagittal-Plane Motion 19
2.4.5 Balance Controller for the Frontal-Plane Motion 21
2.4.6 Computation of the Desired COP 22
2.5 System Measurement 24
2.5.1 COM Measurement 25
2.5.2 COP Measurement 25
2.5.3 Configuration Measurement of the Simplified Models 27
2.6 Full-Body Mapping and Simulation 28
2.6.1 Torque Mapping 28
2.6.2 Simulation 31
2.7 Experiments on the Sarcos Humanoid Robot 32
2.8 Conclusions and Future Work 34
3 MANIPULATING A DYNAMIC OBJECT BY ACTIVE WALKING 36
3.1 Introduction 36
3.1.1 Main Results 36
3.1.2 Organization 37
3.2 Previous Work 37
3.2.1 Biped Locomotion Generation 37

3.2.2 Limit Cycle Walking of Passive Walkers 38
3.3 Static Walking Gait Generation 39
3.3.1 Frontal Motion Planner 39
3.3.2 Sagittal Motion Planner 41
3.3.3 Simulation Results 44
3.4 Cyclic Walking Gait Generation 45
3.4.1 Equation of Motion 46
3.4.2 Collision Model 48
3.4.3 Computing a Cyclic Walking Gait 52
3.4.3.1 Cost Function of the Optimization 52
3.4.3.2 Constraints of the Optimization 55
3.4.4 Maintaining a Cyclic Walking Gait 56
3.4.4.1 Inverse Kinematics 56
3.4.4.2 Initial State for the Next Step 57
3.4.5 Simulation Results 59
3.4.5.1 Setup for Optimization 59
3.4.5.2 Optimal Cyclic Gait 59
3.4.5.3 Simulation Under Disturbance 61
3.5 Conclusions and Future Work 62
4 MOTION TRACKING CONSIDERING STRICT CONTACT CONSTRAINTS .. 66
4.1 Introduction 66
4.1.1 Main Results 67
4.1.2 Organization 68
4.2 Previous Work 68
4.3 Full-Body Dynamics of a Humanoid Robot 69
4.4 Motion Tracking Controller 72

4.4.1 Proportional-Derivative (PD) Controller 73
4.4.2 Joint Torque Optimization Module 74
4.4.2.1 Step 1—Computing contact forces 75
4.4.2.2 Step 2—Computing joint torques 76
4.5 Computation of Contact Forces 78
4.5.1 Basic Formulations 79
4.5.2 Computing Feasible Contact Forces 83
4.5.3 Computing Minimum Contact Forces 86
4.6 Simulation 90
4.6.1 Simulation Setup 90
4.6.2 Tracking Human Motion Without Contact State Change 90
4.6.3 Tracking Human Motion With Contact State Change 92
4.6.4 Tracking Extreme Reference Motion 93
4.7 Conclusions and Future Work 95
5 CONCLUSION 98
BIBLIOGRAPHY 100

LIST OF TABLES

2.1 Common Sensors used on a Robot and Their Functions 24

2.2 Measurement of Quantities 28

LIST OF FIGURES

2.1 Sarcos humanoid robot standing on a free-rolling cylinder. The 3-D motion of the robot with the cylinder is decomposed into two 2-D motions in the sagittal and frontal planes, for which different simplified models are used in designing balance controllers. 14

2.2 Simplified dynamics model for a biped robot with (a) flat feet or (b) geta feet on a rolling cylinder in the sagittal plane. 15

2.3 Control framework. 16

2.4 Simplified dynamics model of a biped robot with flat feet on a rolling cylinder in the sagittal plane. (a) θ_0 is the rolling angle of the cylinder, which also indicates the relative position of the ankle joint to the top of the cylinder if the ankle joint is initially above the top. (b) θ_1 is the rolling angle of the foot relative to the cylinder. (c) θ_2 is the angle of the ankle joint, indicating the center of mass (COM) position relative to the ankle. (d) l is the change of the distance. 16

2.5 Measurement of the actual COP and the state of (a) the flat-feet model or (b) the geta-feet model. 25

2.6 Simulation of a biped robot with (a) flat feet or (b) geta feet on a rolling cylinder. The blue arrow represents the external disturbance applied to the robot. 32

2.7 Experimental setup on the Sarcos humanoid robot. 33

2.8 Conversion of the desired COP to the foot rotation. (a) The desired COP moving forward. (b) Increasing the ankle torque in the direction as shown. (c) Extending the feet. 34

3.1 Framework for walking motion generation on a rolling cylinder. 40

3.2 Framework for sagittal walking motion generation on a rolling cylinder. 41

3.3 Illustration of the motion of the swing foot in a step. The swing foot (in the blue color) (a) rotates together with the supporting foot (not shown) and rolls the cylinder for angle α, (b) lifts up, (c) moves backwards, and then (d) touches down. Once touching down, the swing foot becomes the supporting foot (in the green color) for the next step and (e) rotates and rolls the cylinder for another angle α. (a) and (e) happen at the same time in one step on the swing foot and the supporting foot, respectively. 43

3.4 Snapshots of biped walking on a rolling cylinder. (a) Initial pose. (b) Moving the weight to the left foot and lifting up the right foot. (c) Touching down the right foot at a position behind the left foot. (d) Rotating the cylinder forward while moving the weight to the right. (e)-(g) Repeating this procedure with the right foot as the supporting foot and the left foot as the swing foot. 45

3.5 Graphs of (a,b) the reference and actual COM trajectories and (c,d) the desired, optimized, and actual COP trajectories. 46

3.6 Collision model. The motion of the cylinder is described using variables x_o, y_o, θ_0, while that of each leg is described using variables $x, y, \alpha, \theta_2, l$. The subscripts c and s represent the colliding (swing) leg and the supporting leg, respectively. 47

3.7 Distribution of optimal initial states in the state space. Blue and red dots represent the optimal initial sates with smaller and larger values of the cost function defined by (3.32). 60

3.8 Distribution of optimal initial states with smaller (marked in green) and larger (marked in red) energy consumption for one step. 60

3.9 Optimal initial states colored according to the change in the cost function while shifting optimal initial states in the normal direction. The yellow (red) color means that the change is relatively smaller (bigger). .. 61

3.10 One hundred walking cycles starting with an optimal initial state. Each curve from a dot to a square represents a step. The red color denotes the first step starting with the optimal initial state, while the blue color denotes the last step. 62

3.11 Walking cycles under disturbances in the model and the ankle torque. The red color denotes the first step starting with the optimal initial state, while the green and blue colors denote the last two steps, which are slightly different due to the random disturbance. 63

3.12 Walking cycles with desired average velocity equal to $0.4\ \mathrm{rad/s}$. 64

3.13 Snapshots of one cyclic step with average velocity equal to (a) $0.2\ \mathrm{rad/s}$ and (b) $0.4\ \mathrm{rad/s}$ under different disturbances. 65

4.1 Example of human motion tracking. (a) The original human motion. (b) The simulated robot motion. 67

4.2 Illustration of the relationship between the contact wrench applied to a contact link and the contact forces from the environment. 71

4.3 Overview of the motion tracking controller consisting of two main steps in the dashed box: 1) computation of the contact wrenches to realize the desired floating-base motion, and 2) computation of the joint torques to realize the desired full-body motion. The symbols are defined in the text. 73

4.4 Illustration of the algorithm in 3-D space to compute the minimum distance between $\boldsymbol{w}$ and $V = \mathrm{CO}(W)$. By iteration it generates a sequence of simplicial cones $\mathrm{CO}(V_k)$ in the convex cone $\mathrm{CO}(W)$ such that the point $\boldsymbol{v}_k$ in $\mathrm{CO}(V_k)$ having the minimum Euclidean distance to the point $\boldsymbol{w}$ converges to the closest point in $\mathrm{CO}(W)$ to $\boldsymbol{w}$. Then, the distance between them gives the Euclidean separation distance between $\mathrm{CO}(W)$ and $\boldsymbol{w}$. 83

4.5 Illustration of two iteration strategies to compute the minimum contact forces. (a) A subset V_{k+1} of $W - \boldsymbol{z}_k$ is computed at every iteration by **Algorithm 1** such that $\hat{\boldsymbol{w}}$ is a positive combination of V_{k+1}. Then, $\boldsymbol{z}_{k+1}$, a convex combination of $Z_{k+1} = V_{k+1} + \boldsymbol{z}_k$ that is proportional to $\hat{\boldsymbol{w}}$, is closer to $\boldsymbol{z}$ than $\boldsymbol{z}_k$. Hence, $\boldsymbol{z}_k$ approaches $\boldsymbol{z}$ as the iteration proceeds. (b) $\mathrm{CH}(Z)_k$ (red triangle) is a facet in W such that its interior contains a point $\boldsymbol{z}_k$ proportional to $\hat{\boldsymbol{w}}$ and $\boldsymbol{n}_k$ is the normal of the facet such that $\boldsymbol{n}_k^T \hat{\boldsymbol{w}} > 0$. Using a subset of the vertices of $\mathrm{CH}(Z)_k$ and $\boldsymbol{s}_W(\boldsymbol{n}_k)$, I can find a new facet (green triangle) that contains a new point $\boldsymbol{z}_{k+1}$ in the direction $\hat{\boldsymbol{w}}$ strictly closer to $\boldsymbol{z}$ than $\boldsymbol{b}_k$. This iteration can proceed until $\boldsymbol{z}_k$ converges to $\boldsymbol{z}$, provided that $\boldsymbol{z}_{k+1}$ always lies in the interior of the new facet. 87

4.6 Example of (a) original and (b) simulated motion of "I'm a little teapot" performed by subject 1. 91

4.7 Example of (a) original and (b) simulated motion of "I'm a little teapot" performed by subject 2. 92

4.8 Optimized (red) and actual (blue) contact forces and moments on the left foot. 93

4.9 Simulated side stepping motion. 94

4.10 Simulated forward stepping motion. 95

4.11 Optimized (red) and actual (blue) contact forces in x- and z-directions and moments in y-direction on the left (upper) and right (below) feet for tracking the stepping motion. 96

4.12 Example of preventing falling in tracking extreme motions. (a) Unbalanced reference pose for tracking (transparent) and static equilibrium pose for falling prevention (opaque). (b) Final pose of the robot, which is close to but does not reach the tracking reference pose. (c) The contact forces at the local COP on each foot (white lines) and the total contact force at the global COP (red line), which reaches the boundary of the support area. 97

LIST OF ABBREVIATIONS

COM	Center of Mass
COP	Center of Pressure
DOF	Degree of Freedom
IMU	Inertial Measurement Unit
ZMP	Zero Moment Point

CHAPTER 1: INTRODUCTION

Robots have the potential to significantly change human life in the 21st century. Robotics research is essential in bringing robots to benefit our daily lives. Of particular interest is research into humanoid robotics, whose ultimate goal is to build a robot that not only looks but also behaves like a human, and may even become more capable than a human. To realize this goal, a humanoid robot needs first to be able to conduct the tasks fundamental to legged robots, such as balancing and walking. However, simply performing such basic tasks in static, open environments is far from enough for a robot to be useful. A humanoid robot should also possess the ability to make use of external tools or objects, such as a bicycle and a skateboard, to generate dynamic motions and improve its mobility. Then, the ability to manipulate or maneuver external tools or objects must be considered together with the locomotion of the robot itself. Also, humanoid robots are expected to work and live closely with humans, so having human-like motions is important for them to be human-friendly and finally merge into the human society. In some applications, such as performing a drama and conducting an orchestra on the stage, a humanoid robot must be capable of providing stylized human-like motions. In all such scenarios, a humanoid robot needs to not only perform a single basic locomotion task, such as balancing or walking, but also achieve other goals, such as manipulating an object or imitating human motions.

This dissertation proposes new techniques to improve the capability and the quality of locomotion of humanoid robots by considering additional kinematic and dynamic constraints. First, I enhance the mobility of a humanoid robot by endowing it with the ability to manipulate a dynamic object. Taking a free-rolling cylinder as a study case, I investigate how to enable a humanoid robot to maintain balance in a highly dynamic environment by actively manipulating the environment while generating a bipedal walking behavior. This

locomotion task is challenging even for normal humans. There are many problems that need to be solved to make it happen on a humanoid robot. For example, one essential problem is how to design controllers to control not only the motion of the robot but also the cylinder's motion. A further problem is what walking gait the robot should and can perform on top of the cylinder in order to roll the cylinder. In tackling these problems, many factors need to be considered, such as the dynamics of the robot and the cylinder, the impact when the swing foot leaves or touches the cylinder, various physical and dimensional constraints, disturbances and errors that may cause the robot to deviate from a planned gait, and so on.

Second, I present a method for humanoid robots to track given reference motions, such as captured human motions or motions defined by key frame animation, such that they can have human-like or stylized motions. The motion of a floating-base humanoid robot depends not only on the joint torques but also the reaction contact forces from the environment, which are dependent on the applied joint torques and subject to the nonlinear friction constraints. Direct proportional-derivative (PD) tracking usually does not work in this case because the desired joint accelerations from traditional PD control law are not necessarily achievable due to the limitation in friction. Also, the contact links of the robot in a reference motion usually do not comply with the current environment and a kinematic constraint needs to be imposed on each of them to correct their motions. Therefore, how to take into account these constraints in motion tracking, while keeping the computation fast enough for real-time control, is a challenging problem.

1.1 Locomotion Generation Overview

The term "locomotion" in robotics is the collective name for various types of motions that a robot uses to move itself or part of its body in the environment. Typical types of motions for humanoid robots include balancing, walking, running, etc. For different types of motions, the motion generation and control methods will be different. Nevertheless, some

common information and components are required by almost all the methods. In this section, I provide a high-level introduction.

1.1.1 Environmental Property

Locomotion generation must take into account the environmental property for a generated motion to be consistent with and physically achievable in the environment. First, the geometric information of the environment needs to be considered. For example, to generate a walking behavior, we need to know whether it happens on a horizontal ground, slopes, stairs, or even rough terrains, so that specific motion strategies can be used. Second, we need to know some physical parameters, such as the friction coefficient. If the robot is required to interact with some dynamic objects in the environment, then the kinematic and dynamic properties of the objects should be known as well.

1.1.2 Robot Model

In order to generate a motion that a robot can carry out, the model of the robot, including its kinematic and dynamic parameters, must be known. The kinematic parameters include the structure (the number and types of joints and their arrangement) and the dimension (the size of each link) of the robot, which are needed in both kinematics and dynamics calculations. The dynamic parameters include the mass and inertia and the position of the center of mass of each link, which are used together with the kinematic parameters in the forward/inverse dynamics calculation.

1.1.3 Sensor Information

Most of robots use closed-loop control, which means that the control command changes based on the difference between the actual and desired states of the robot. A humanoid robot is usually equipped with several basic sensors to measure its actual state. First, an inertial measurement unit (IMU) is often attached to the floating base of a humanoid robot to

measure its orientation and acceleration. Second, force-torque sensors are common sensors installed on robot's feet to measure the contact forces applied to the feet by the environment. Using the force-torque sensor data, we can also estimate the position of the center of pressure (COP), which plays an important role in the balance control of a humanoid robot. Third, joint angle sensors are usually embedded in every joint to record the joint angle and velocity. Those data are required by many components in the locomotion generation and control, such as the design of controllers and the calculation of the full-body dynamics of the robot. Depending on the requirement of a locomotion task, other sensors or sensing techniques may be used as well, such as vision sensors and ultrasonic sensors. In my work, I only use the aforementioned three conventional sensors.

1.1.4 Framework for Motion Generation and Verification

At a high level, I would like to explain the control loop for humanoid locomotion generation, which usually consists of a control unit and a simulation or hardware execution unit. The control unit is responsible for the balance control and the production of control command for the robot to perform a desired motion. The balance control is usually based on a simplified dynamics model of the robot, such as the inverted pendulum model. Using the simplified model, a feedback controller can be built to generate some high-level control information, such as the desired COP position for the robot to track so as to maintain balance, based on the actual state and the desired state of the simplified mode. The state of the model refers to both its configuration and velocity. While the actual configuration of the model can be directly estimated based on the sensor information of the robot, there is no direct access to the actual velocity of the model. Also, sensor information is often noisy. These problems can be solved by adding an observer, which filters out the sensor noise and estimates the actual velocity based on the estimated configuration. After the high-level control information is produced, it needs to be converted to the control command for the robot base on its full-body dynamics. Depending on how the robot is controlled, the control command can be joint

angles for position-controlled robots or joint torques for torque-controlled robots, and the ways to perform this conversion are different.

The robot in simulation or hardware accepts the control command and should produce the planned motion if the designed controllers work properly. However, it is inevitable that the model used to compute the control command has some difference from the real robot. Due to the modeling error, the robot usually does not generate the exactly same motion as the planed one and a difference always exists between them. Then, the controllers are expected to capture this difference and consider it in the computation of the control command, typically through feedback, so as to correct the generated motion.

In this dissertation, I focus on the control unit, including balance control, walking gait generation, and motion tracking control.

1.2 Locomotion of Humanoid Robots

Nowadays, many robots are capable of basic locomotion tasks, such as walking and running (Takanishi et al., 1988; Kajita et al., 2005), and many methods for locomotion generation and control of humanoid robots have been proposed. The following subsections discuss well-known approaches that are closely related to the work presented in this dissertation.

1.2.1 Robot Balance Control

Balancing on its own is the most fundamental locomotion task that a robot should be capable of. The balance control of a humanoid robot often uses the concept of zero moment point (ZMP), also known as the center of pressure (COP), and requires the ZMP to stay within the contact region provided by the robot feet, which is a common feasibility criterion used in generating humanoid motions. While a simplified model, such as an inverted pendulum (Kajita and Tanie, 1995; Kajita et al., 2001), is often used to characterize the dynamics of the robot, a feedback controller can be designed based on the simplified

model to produce the desired positions of the ZMP and the center of mass (COM) for the robot to maintain balance. The control command for the robot to generate the full-body motion and realize those positions, namely joint angles for position-control robots or joint torques for torque-controlled robots, is computed through inverse kinematics or inverse dynamics (Kajita et al., 2003; Yamane and Hodgins, 2009). One major issue with this method is that the ZMP-based feasibility criterion is only applicable to horizontal terrains. Ott et al. (2011) used a more general criterion based on feasible contact forces and developed a balance controller that can be applied to any terrains.

The above work assumes that the robot is in a stationary environment. So far, the study of robot balance control in non-stationary environments is still limited. Kuroki et al. (2003; 2004) proposed a motion control system to maintain balance of a small biped robot on a moving platform or under external forces. For the same purpose, Hyon (2009) presented a contact force control framework for the balance control of a human-size robot on rough terrains under external forces, while Anderson and Hodgins (2010) developed methods for adapting models of humanoid robots performing dynamic tasks. However, these control techniques only allow robot to passively adapt to the environmental change and cannot be used in the balance control by manipulating a dynamic object as I do in this dissertation.

1.2.2 Biped Locomotion Generation

The generation of walking patterns for a biped robot often assumes that the footsteps are given, and the problem becomes how to compute the joint angles such that the resulting motion satisfies the balance condition and the given footsteps. Based on the given footsteps, one may first determine a ZMP or COP trajectory and compute a physically consistent COM trajectory of the robot using a simplified model, such as an inverted pendulum (Kajita and Tanie, 1995; Kajita et al., 2001). Then, the joint angles of the robot can be calculated using inverse kinematics, according to the COM trajectory and the footsteps. This technique has been successfully used to generate biped walking patterns in a stationary environment (Sug-

ihara et al., 2002; Kajita et al., 2003). In an environment with known obstacles, several algorithms have been proposed to compute footsteps and a collision-free path for a biped robot (Kuffner et al., 2001; Chestnutt et al., 2005; Ayaz et al., 2009), such that the walking pattern can be generated later in this way. Some work tried to mimic human walking motion by adding single toe support phase and characterized swing leg motions (Miura et al., 2011). To avoid using the ZMP feasibility criterion and generate biped walking on uneven terrain, Hirukawa et al. (2006) proposed a walking pattern generator using a general criterion based on feasible contact forces.

Humanoid locomotion can also be generated in a reverse way. One may first generate a reference COM or upper-body motion for the robot. Then, the required COP trajectory is computed from the reference reference motion based on a simplified model, and appropriate footsteps that cover the computed COP trajectory are also determined. Finally, full-body trajectories in terms of joint angles of the robot can be calculated likewise through inverse kinematics. Unfortunately, there is not much work on this kind of methods (Sugihara, 2008).

The above walking generation methods consider the robot to walk on the ground that does not move. Hence, they cannot be used in my case where a robot is required to walk on a dynamic object and manipulate the object by bipedal walking.

1.2.3 Motion Tracking Control

Since humanoid robots have structures similar to humans, using human motion capture data to program humanoid robots seems to be an effective way to generate human-like motions. The work (Ude et al., 2000; Safonova et al., 2003) mapped human motions to fixed-base humanoid robots considering the kinematic constraints of the robot. Adapting human motion data to the dynamics of floating-base humanoid robots was discussed in the work (Ikemata et al., 1999; Yamane and Nakamura, 2003). Miura et al. (2009) and Boutin et al. (2010) developed methods for generating humanoid locomotion based on motion capture data, which modify the extracted joint trajectories according to a replanned ZMP

trajectory that ensures the dynamic consistency. Nakaoka et al. (2003) proposed a method to convert human dancing motions to physically feasible motions for humanoid robots by manually segmenting a motion into motion primitives and designing a controller for each of them. However, these approaches are aimed at offline planning.

Some methods can realize online tracking of upper-body motions in the double-support phase while using the lower body for balancing (Zordan and Hodgins, 2002; Ott et al., 2008). Yamane and Hodgins (2009; 2010) presented controllers for humanoid robots to simultaneously track motion capture data and maintain balance. However, most of the previous work omitted the friction constraint on contact forces by assuming that the friction coefficient is big enough or considered it as a penalty term in the computation to simplify the problem. As a consequence, the generated motion may cause the violation of the friction constraint and not be achievable in practice.

Using human motion data to generate motion for humanoid characters has also been studied in computer graphics (Tak et al., 2000; Safonova et al., 2004; Sok et al., 2007; da Silva et al., 2008; Muico et al., 2009). Nevertheless, those approaches usually employ an extensive optimization process and cannot be applied to realtime control of humanoid robots.

Developing efficient methods for realtime motion tracking for humanoid robots considering strict friction and other constraints still remains to be a difficult problem.

1.3 Thesis Statement

Considering dynamic and kinematic constraints in the environment in the controller design enables humanoid robots to enhance locomotion tasks by manipulating a dynamic object or tracking given reference motions, while maintaining balance.

1.4 Main Results

In support of my thesis statement, I present several major results on generating dynamic, human-like motions on humanoid robots. First, I demonstrate how to design a balance controller such that a humanoid robot maintains balance on an object that will move because of the interaction with the robot. Second, I discuss how to actively manipulate such a dynamic object and control its motion by generating walking behaviors on top of it to improve the mobility of the robot itself. Finally, I provide a method for enabling humanoid robots to produce stylized motions using motion capture data.

1.4.1 Balance Control in a Dynamic Environment

One important result included in this dissertation is a balance control technique for a humanoid robot having interaction with dynamic objects in the environment. Taking a cylinder that rolls freely on the ground as an example, I present a balance controller such that a humanoid robot can balance on the cylinder under disturbances. In order for the robot to actively maneuver cylinder's motion rather than passively adapt to it, I take into account the dynamics of the cylinder together with that of the robot in the balance controller design. This technique can potentially be applied to the balance control of humanoid robots in other similar scenarios, such as riding a unicycle or pushing a wheelbarrow, as long as the dynamics of the object or the tool that the robot operates can be modelled. Using the proposed balance controller, I enable a real robot, the Sarcos humanoid robot, to balance on a free-rolling cylinder. These results will be discussed in Chapter 2.

1.4.2 Manipulating a Dynamic Object by Active Walking

Beyond balance control, I present a method for generating biped walking behaviors of a humanoid robot using a dynamic object. Still taking the rolling cylinder as an example, I use this method and compute the gait for the robot to walk on top of the cylinder in a cyclic

pattern and roll the cylinder at a desired speed. A collision model between the swing leg and the cylinder is derived to compute the velocity change of the robot and the cylinder when the swing leg touches down. I also present a method for the robot to maintain a planned gait under disturbances. These results will be addressed in Chapter 3.

1.4.3 Motion Tracking under Strict Contact Constraints

Additionally, in this dissertation I present a motion tracking control method for humanoid robots to reproduce given reference motions, such as motions defined by human motion capture data or key frame animation. The method considers the exact full-body dynamics of the robot and the strict friction constraint on contact forces. By taking advantage of the property that the joint torques do not contribute to the six DOFs of the floating base of the robot, I separate the computation of contact forces from joint torques. As a result, computing the contact force that satisfies the friction constraint can be reduced to a simple quadratic program with second-order cone constraints, for which I present several efficient algorithms. Once the contact forces are determined, the optimal joint torques can be computed in a closed form. While the solution satisfies strict friction constraints, this method is fast enough for realtime motion tracking and can be applied to motions on uneven terrains or involving contacts at hands or links other than robot's feet. These results will be discussed in Chapter 4.

Finally, Chapter 5 provides a summary of my contributions and a discussion of future work.

CHAPTER 2: BALANCE CONTROL ON A DYNAMIC OBJECT

2.1 Introduction

Most of the previous work on robot locomotion assumes a stationary environment and considers only the motion of the robot itself. For a robot to be truly useful for humans, however, it should also be able to manipulate objects and make changes in the environment while performing a basic locomotion task. Such capability is often required in our daily activities, such as pushing a cart while wandering in a supermarket. In this and the next chapters I discuss how I realize a generic biped humanoid robot that actively manipulates an object in the environment and performs a dynamic motion. In this particular piece of work, I choose walking on a free-rolling cylinder as an example of such behaviors, which is difficult even for normal humans. Considering cylinder's movement is mandatory because the interaction with the environment is limited to the contact between the cylinder and floor. This chapter introduces the development of a balance controller that enables a humanoid robot to stand and maintain balance on the cylinder. The balance controller provides the basis for generating biped walking behaviors of the robot on the cylinder, which will be discussed in the next chapter.

2.1.1 Main Results

This chapter describes the balance control of a humanoid robot standing on a cylinder that can roll freely on the horizonal ground. Based on a simplified model I first design a state-feedback balance controller that produces the desired COP position for the robot to follow so as to maintain balance on the cylinder. A tracking controller is applied to computing joint torques for the full-body control of the robot to realize the desired COP.

In order for the robot to actively maneuver cylinder's motion rather than passively adapt to it, I take into account the dynamics of the cylinder together with robot's own dynamics in the development of the balance controller and the tracking controller. Both simulation and hardware experimental results are provided at the end of this chapter to show the effectiveness of this balance control method.

2.1.2 Organization

This chapter is organized as follows. Section 2.2 briefly reviews the related work. Section 2.3 gives an overview of my work on balance control. Section 2.4 describes the simplified model and the details of the balance controller. Section 2.5 discusses available sensors and system measurement based on sensor data. Section 2.6 details the joint torque computation and full-body simulation. Section 2.7 reports the experiment on a real humanoid robot. In Section 2.8, I discuss the limitation and the possible extension of this work.

2.2 Previous Work

One of the most popular methods for the balance control of humanoid robots is designing a feedback controller based on a simplified model, such as a typical inverted pendulum (Kajita and Tanie, 1995; Kajita et al., 2001). The feedback controller usually determines the desired positions of the ZMP or COP and the COM for the robot to maintain balance. The joint angles or joint torques that are required to generate the full-body motion of the robot and realize the desired positions are computed through inverse kinematics or inverse dynamics, depending on what control command the robot accepts (Kajita et al., 2003; Yamane and Hodgins, 2009). One major issue with this method is that the ZMP feasibility criterion is only applicable to horizontal terrains. Ott et al. (2011) used a more general criterion based on feasible contact forces in developing a balance controller that is applicable to any terrains.

So far, the study of robot motion control in non-stationary environments is still limited. Kuroki et al. (2003; 2004) proposed a motion control system to maintain balance of a small biped robot on a moving platform or under external forces. For the same purpose, Hyon (2009) presented a contact force control framework for the balance control of a human-size robot on rough terrains under external forces, while Anderson and Hodgins (2010) developed methods for adapting models of humanoid robots performing dynamic tasks. In these cases, however, the robot's feet keep stationary contact with the platform or ground and no stepping motion is involved.

Besides humanoid robots, some other robots may work in a non-stationary environment, such as a singe spherical wheeled mobile robot (Nagarajan et al., 2009, 2013) and multi-wheeled robots balancing on and driving a ball (Endo and Nakamura, 2005; Lauwers et al., 2006; Kumagai and Ochiai, 2009). In that case, the wheels always make three or four symmetric contacts with the ball, which greatly benefits the balance control of the robot. In my case, however, the feet of a biped robot can only make one or two contacts with a cylinder, which are usually asymmetrical about the top of the cylinder. Furthermore, because of the limited foot size and support region, the ideal COP, which is continuously changing on a rolling cylinder, may go beyond the support region. Hence, I have to not only design controllers to maintain system's balance but also combine them with a stepping motion generator to provide the robot with timely support on the rolling cylinder during walking, which will be discussed in the next chapter.

2.3 Problem Overview

2.3.1 Problem Statement

The goal of this chapter is to realize the balance control of a humanoid robot standing on a free-rolling cylinder, as depicted in the leftmost figure of Fig. 2.1. I have two designs of feet for the robot to do so, i.e., normal flat feet and geta feet inspired by the traditional Japanese footwear called Geta, as shown in Fig. 2.2. A flat foot makes only one edge

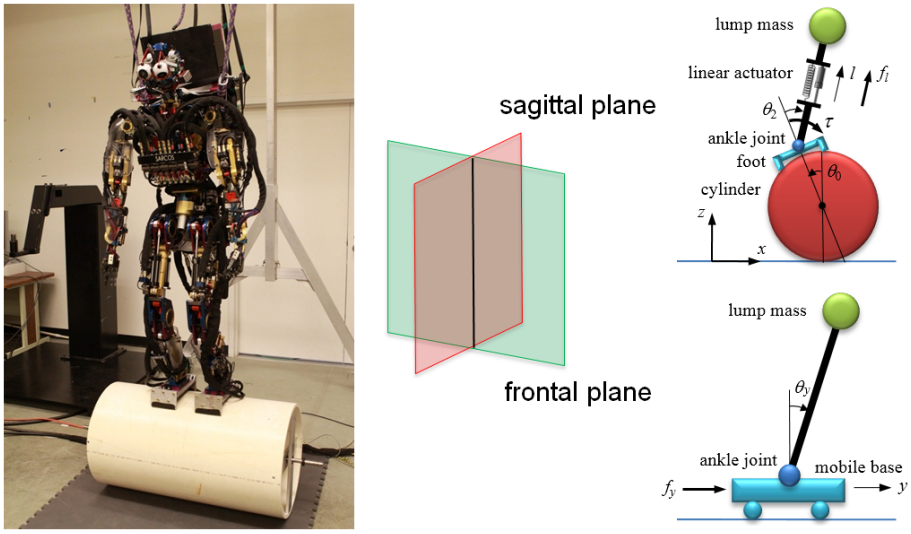
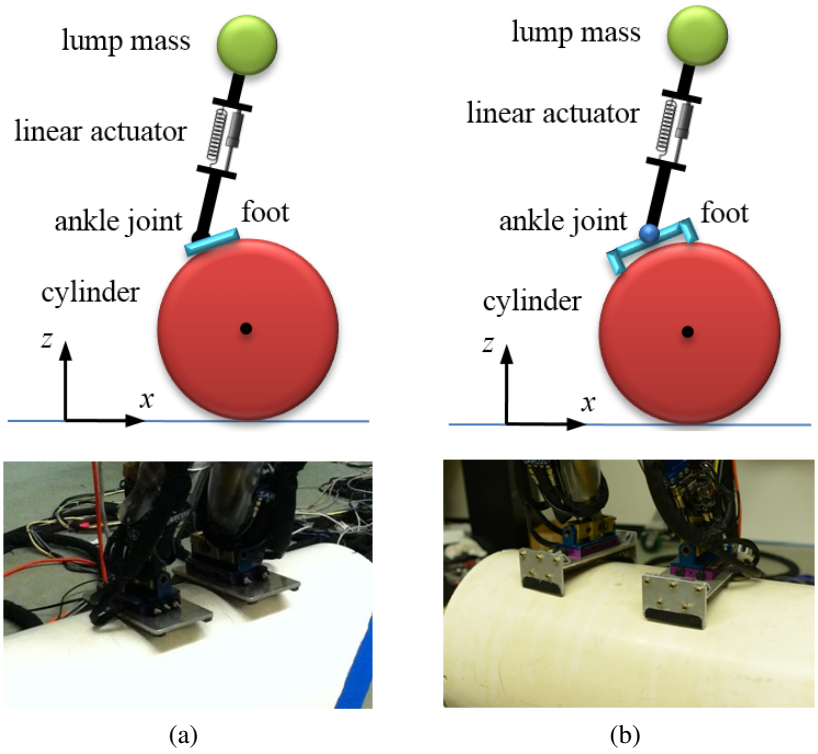


Figure 2.1. Sarcos humanoid robot standing on a free-rolling cylinder. The 3-D motion of the robot with the cylinder is decomposed into two 2-D motions in the sagittal and frontal planes, for which different simplified models are used in designing balance controllers.

contact with the cylinder, so the foot can rotate about the edge relative to the cylinder, which gives the robot more degrees of freedom but makes the balance control more difficult. By contrast, a geta foot, which is built by attaching an extra plate to both toe and heel sides of a normal flat foot, can contact the cylinder by two edges. Then, the geta foot cannot move relative to the cylinder while maintaining the full contact, assuming that it does not slip either. The two contact edges form a much bigger rectangular support region for the robot, which significantly facilitates the balance control.

2.3.2 Control Framework

Fig. 2.3 depicts the entire control framework, which consists of four units, i.e., a balance controller, a full-body mapping unit, a state measurement unit, and the robot. Based on simplified dynamics models of the system including the robot and the cylinder, the balance controller generates the desired COP position as the high-level control information for the robot to maintain balance on the cylinder. Then, the full-body mapping unit determines the full-body control command, consisting of joint torques or angles, for the real robot to realize the desired COP as well as a given reference standing pose on the cylinder. The



(a) (b)

Figure 2.2. Simplified dynamics model for a biped robot with (a) flat feet or (b) geta feet on a rolling cylinder in the sagittal plane.

state measurement unit measures the actual states of the simplified models based on various sensor data from the robot. The actual states are used as a part of the input to the balance controller. In the rest of this chapter, I will discuss each unit in the control framework.

2.4 Balance Controller

Similarly to many previous works, I decompose the 3-D motion of a biped robot on a rolling cylinder into two 2-D motions in the sagittal and frontal planes and design the balance controller for each decomposed motion, as depicted in Fig. 2.1. Since the cylinder can roll in the sagittal plane and the robot is expected to actively maneuver it, I include the cylinder in the sagittal-plane model. However, the cylinder is not included in the frontal-plane model, since it is supposed not to move in the frontal plane. In addition, I use different simplified models for the cases that different foot designs are adopted, as shown in Fig. 2.2. In the

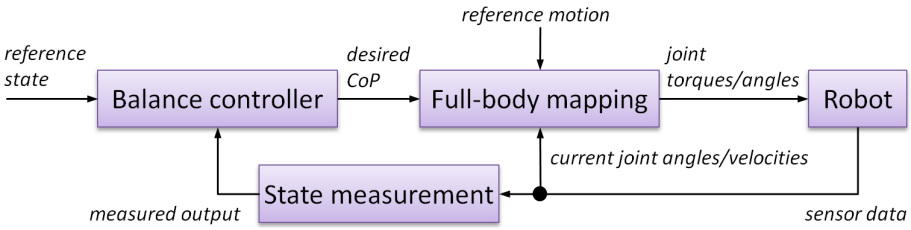


Figure 2.3. Control framework.

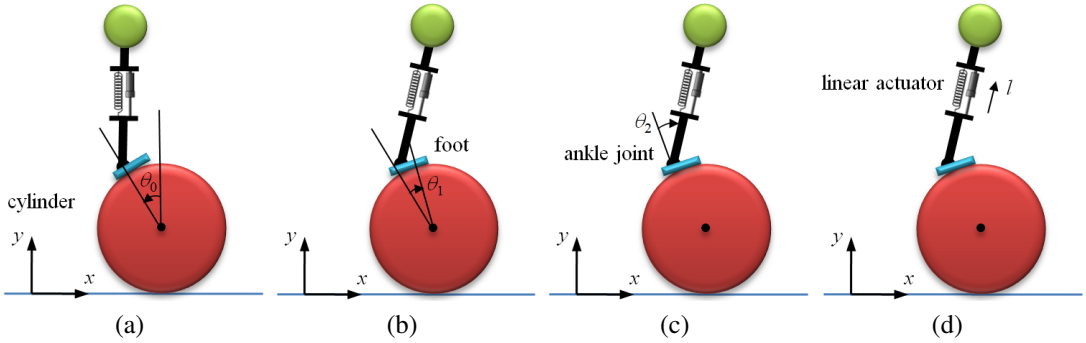


Figure 2.4. Simplified dynamics model of a biped robot with flat feet on a rolling cylinder in the sagittal plane. (a) θ_0 is the rolling angle of the cylinder, which also indicates the relative position of the ankle joint to the top of the cylinder if the ankle joint is initially above the top. (b) θ_1 is the rolling angle of the foot relative to the cylinder. (c) θ_2 is the angle of the ankle joint, indicating the center of mass (COM) position relative to the ankle. (d) l is the change of the distance.

following discussion, I first derive the linearized equation of motion for each simplified model and then discuss how to design the balance controller and calculate the desired COP from the output of the balance controller.

2.4.1 Flat-Foot Sagittal-Plane Simplified Model

I use a simplified model to characterize the dynamics of the whole system, including the robot and the cylinder, and design a balance control for the system based on the simplified model.

Fig. 2.4 depicts the simplified model of a biped robot with flat feet on the cylinder and the kinematic and dynamic parameters of the model. From the bottom up, the sagittal-plane model consists of the cylinder, a foot, an ankle joint, and a lump mass that is connected

to the ankle joint through a link with a linear actuator on it. The linear actuator is used to model the effect of the knee joint. The configuration of the model can be described by three angular variables θ_0, θ_1 and θ_2 and a linear variable l, as indicated in Fig. 2.4, where θ_0 represents the rolling angle of the cylinder (Fig. 2.4a), θ_1 denotes the angle of the foot rotating relatively to the cylinder (Fig. 2.4b), and θ_2 is the angle of the ankle joint and l is the change in the link length (Fig. 2.4c). Assume that there is no slip between the foot and the cylinder. The positive direction of angles is taken to be clockwise in Fig. 2.4. Let r_0, m_0, and I_0 respectively denote the radius, the mass, and the inertia of the cylinder, m_1 and I_1 the mass and the inertia of a foot, m_2 and I_2 the mass and inertia of the inverted pendulum, and $L = l_0 + l$ the distance between the ankle joint and the lump mass, where l_0 is the distance while the robot is in the rest position. The COM of the foot is assumed to be at the ankle.

The simplified model can be used to characterize the dynamics of the robot standing on the cylinder with both legs or just one leg if the swing leg dynamics is ignored. The linearized equation of motion of the model can be derived as

$$\boldsymbol{M}\ddot{\boldsymbol{\theta}} + \boldsymbol{G}\boldsymbol{\theta} = \boldsymbol{\tau} \tag{2.1}$$

where $\boldsymbol{\theta} = [\theta_0 \quad \theta_1 \quad \theta_2 \quad l]^T$, $\boldsymbol{\tau} = [0 \quad 0 \quad \tau \quad f]^T$, τ is the ankle torque, f is the force from the linear actuator and

$$\boldsymbol{M} = \begin{bmatrix} M_1 + I & M_2 + I_1 & M_2 & 0 \\ M_2 + I_1 & M_3 + I_1 & M_3 & 0 \\ M_2 & M_3 & M_3 & 0 \\ 0 & 0 & 0 & m_2 \end{bmatrix}, \quad \boldsymbol{G} = -\begin{bmatrix} G_1 + m_2 g l_0 & m_2 g l_0 & m_2 g l_0 & 0 \\ m_2 g l_0 & m_2 g l_0 - G_1 & m_2 g l_0 & 0 \\ m_2 g l_0 & m_2 g l_0 & m_2 g l_0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_1 = m_0 r_0^2 + 4 m_1 r_0^2 + m_2 L_1^2, \quad M_2 = m_2 l_0 L_1 + I_2, \quad M_3 = m_2 l_0^2 + I_2$$

$$L_1 = 2r_0 + l_0, \quad I = I_0 + I_1 + I_2, \quad G_1 = (m_1 + m_2) g r_0.$$

2.4.2 Geta-Foot Sagittal-Plane Simplified Model

Since it is assumed that the geta foot does not move relatively to the cylinder, the robot can be deemed to be hinged on the cylinder through the ankle joint. Then, the sagittal-plane model has three DOFs in total and one DOF less than the flat-foot model does. The parameters used to describe the motion of the geta-foot model include the rolling angle θ_0 of the cylinder, the angle θ_2 of the ankle joint, and the length l corresponding to the linear actuator, as indicated in Fig. 2.1.

The linearized equation of motion of the geta-foot sagittal-plane model can be written in the same form as (2.1), where $\boldsymbol{\theta} = [\theta_0 \quad \theta_2 \quad l]^T$, $\boldsymbol{\tau} = [0 \quad \tau \quad f]^T$, τ is the ankle torque, f is the linear actuation force and

$$\boldsymbol{M} = \begin{bmatrix} M_1 + I & M_2 & 0 \\ M_2 & M_3 & 0 \\ 0 & 0 & m_2 \end{bmatrix}, \quad \boldsymbol{G} = -\begin{bmatrix} G_1 + m_2 g L_0 & m_2 g L_0 & 0 \\ m_2 g L_0 & m_2 g L_0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$M_1 = m_0 r_0^2 + m_1 (r_0 + h)^2 + m_2 (r_0 + h + l_0)^2$$

$$M_2 = m_2 l_0 (r_0 + h + l_0) + I_2, \quad M_3 = m_2 l_0^2 + I_2$$

$$I = I_0 + I_1 + I_2, \quad G_1 = (m_1 + m_2) g h.$$

2.4.3 Frontal-Plane Simplified Model

Since the cylinder is supposed not to move in the frontal plane, I use a single inverted pendulum model to characterize the motion of the robot alone, as depicted in Fig. 2.1, just like the case that the robot stands on the horizontal ground. Also this model works for either case where flat or geta feet are used. The inverted pendulum model consists of a mobile base representing the COP and a lump mass representing the COM of the robot. The linearized equation of motion of the model is derived as

$$\boldsymbol{M}\ddot{\boldsymbol{\theta}} + \boldsymbol{G}\boldsymbol{\theta} = \boldsymbol{\tau} \tag{2.2}$$

where $\boldsymbol{\theta} = [y \quad \theta_y]^T$, $\boldsymbol{\tau} = [f_y \quad 0]^T$, and

$$\boldsymbol{M} = \begin{bmatrix} m_1 + m_2 & m_2 l_0 \\ m_2 l_0 & m_2 l_0^2 + I \end{bmatrix}, \quad \boldsymbol{G} = - \begin{bmatrix} 0 & 0 \\ 0 & m_2 l_0 g \end{bmatrix}.$$

Here, y is the position of the mobile base, θ_y is the joint angle, f_y is the horizontal input force applied to the mobile base, m_1 is the mass of the mobile base, m_2 and I are the mass and the inertia of the lump mass, and l_0 is the length of the inverted pendulum.

2.4.4 Balance Controller for the Sagittal-Plane Motion

Here I discuss the balance controller design for the sagittal plane based on the aforementioned simplified models. The balance controller for the frontal plane is given in the next subsection. Recall that there are two simplified models with different feet designs for the sagittal-plane motion. However, the derivations of the balance controller are almost the same, except that the matrices and vectors in the derivation have different dimensions because the two models have different numbers of DOF. Thus I use the flat-foot model as an example hereinbelow to demonstrate how to derive the balance controller.

Equation (2.1) can be rewritten as a state-space differential equation

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.3a}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} \tag{2.3b}$$

where $\boldsymbol{x} = [\boldsymbol{\theta}^T \quad \dot{\boldsymbol{\theta}}^T]^T$ is the state, $\boldsymbol{u} = [\tau \quad f]^T$ is the input, and the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are given by

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0}_{4\times4} & \boldsymbol{I}_{4\times4} \\ -\boldsymbol{M}^{-1}\boldsymbol{G} & \boldsymbol{0}_{4\times4} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0}_{4\times2} \\ \boldsymbol{M}^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_{4\times4} & \boldsymbol{0}_{4\times4} \end{bmatrix}.$$

Because there is no direct access to the real state of the simplified model, I design an observer to estimate it by comparing the estimated output $\hat{\boldsymbol{y}}$ and the measured output $\boldsymbol{y}$:

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{F}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{2.4}$$

where $\boldsymbol{F} \in \mathbb{R}^{8\times 4}$ is the observer gain, the estimated output $\hat{\boldsymbol{y}} = \boldsymbol{C}\hat{\boldsymbol{x}}$, and $\hat{\boldsymbol{x}}$ is the estimated state. The measured output $\boldsymbol{y}$ is obtained from the sensor data, which will be discussed in Section 2.5.3.

Referring to the previous work (Yamane and Hodgins, 2009; Zheng and Yamane, 2011, 2013a), I design a state-feedback balance controller as

$$\boldsymbol{u} = \boldsymbol{K}(\boldsymbol{x}^* - \hat{\boldsymbol{x}}) \tag{2.5}$$

where $\boldsymbol{K} \in \mathbb{R}^{2\times 8}$ is a feedback gain and $\boldsymbol{x}^*$ is an equilibrium state such that $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{0}$. The feedback gain $\boldsymbol{K}$ is chosen so that it ensures that all the eigenvalues of $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}$ have negative real parts and the system asymptotically converges to the equilibrium state. The first and second rows of $\boldsymbol{K}$ contain the feedback gains for generating the ankle torque τ and the linear actuation force f, respectively. Since $\boldsymbol{A}$ here has full rank, $\boldsymbol{x}^* = \boldsymbol{0}$ is the only equilibrium state of the simplified model.

Combining (2.3)–(2.5), I obtain the following system of the estimated state $\hat{\boldsymbol{x}}$ and the new input $\boldsymbol{u}_s = \boldsymbol{y}$:

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}_s\hat{\boldsymbol{x}} + \boldsymbol{B}_s\boldsymbol{u}_s \tag{2.6a}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{C}_s\hat{\boldsymbol{x}} \tag{2.6b}$$

where

$$\boldsymbol{A}_s = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} - \boldsymbol{F}\boldsymbol{C},\ \ \boldsymbol{B}_s = \boldsymbol{F},\ \ \boldsymbol{C}_s = \boldsymbol{C}.$$

Later in Section 2.4.6, I will discuss how to calculate the desired COP based on $\hat{\boldsymbol{y}}$ and/or $\dot{\hat{\boldsymbol{x}}}$.

2.4.5 Balance Controller for the Frontal-Plane Motion

Rewriting (2.2) in the state space yields

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.7a}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} \tag{2.7b}$$

where $\boldsymbol{x} = [y \quad \theta_y \quad \dot{y} \quad \dot{\theta}_y]^T$ is the state, $\boldsymbol{u} = f_y$ is the input, $\boldsymbol{y} = [y \quad \theta_y]^T$ is the output, and the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are given by

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_{2\times 2} \\ -\boldsymbol{M}^{-1}\boldsymbol{G} & \boldsymbol{0}_{2\times 2} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0}_{2\times 1} \\ \boldsymbol{M}^{-1}\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right] \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 2} \end{bmatrix}.$$

The observer is given by

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{F}(\boldsymbol{U}[y_{\mathrm{COP}} \quad y_{\mathrm{COM}}]^T - \hat{\boldsymbol{y}}) \tag{2.8}$$

where $\boldsymbol{U}$ is the matrix given below to convert the measured COP position y_{COP} and COM position y_{COM} to the configuration of the simplified model,

$$\boldsymbol{U} = \begin{bmatrix} 1 & 0 \\ -1/l & 1/l \end{bmatrix}.$$

The measurement of the COP and COM positions will be discussed in Section 2.5.

The design of feedback controller is similar to (2.5). Different from the sagittal-plane model as depicted in Fig. 2.2, the inverted pendulum has an infinite number of equilibrium points, which correspond to any case where the COP and the COM of the robot are on the same vertical line. Hence, the feedback controller for the frontal plane is designed as

$$\boldsymbol{u} = \boldsymbol{K}(\boldsymbol{T}y_{\mathrm{ref}} - \boldsymbol{x}) \tag{2.9}$$

where y_{ref} is the reference COM position and $\boldsymbol{T} = [1 \quad 0 \quad 0 \quad 0]^T$ maps y_{ref} to the corresponding equilibrium state of the inverted pendulum model.

Combining (2.7)–(2.8), I obtain the following system of the estimated state $\hat{\boldsymbol{x}}$ and input $\boldsymbol{u}_f = [y_{\text{COP}} \quad y_{\text{COM}} \quad y_{\text{ref}}]^T$:

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}_f \hat{\boldsymbol{x}} + \boldsymbol{B}_f \boldsymbol{u}_f \tag{2.10a}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{C}_f \hat{\boldsymbol{x}} \tag{2.10b}$$

where

$$\boldsymbol{A}_f = \boldsymbol{A} - \boldsymbol{BK} - \boldsymbol{FC}, \;\; \boldsymbol{B}_f = [\boldsymbol{FU} \quad \boldsymbol{BKT}], \;\; \boldsymbol{C}_f = \boldsymbol{C}.$$

2.4.6 Computation of the Desired COP

From the output of the balance controllers I derive the desired COP, which provides high-level control information and will be used in generating full-body control commands for the robot to maintain balance on the cylinder, as discussed later in Sections 2.6 and 2.7. The y-coordinate (position in the frontal plane) of the desired COP is generated directly in the output of the frontal-plane balance controller. Here I discuss the derivation of the x- and z-coordinates (position in the sagittal plane) of the desired COP from the output of the sagittal-plane balance controller.

The computation of the desired COP in the sagittal plane depends on the adopted sagittal-plane simplified model, i.e., the flat-foot model or the feta-foot model. I start with the case that the simplified model has a flat foot. In this case, the COP is the contact between the foot and the cylinder.

$$x_{\text{COP}} = x_0 + r_0 \hat{\theta}_1 \cos(\theta_0 + \theta_1) \tag{2.11a}$$

$$z_{\text{COP}} = z_0 - r_0 \hat{\theta}_1 \sin(\theta_0 + \theta_1) \tag{2.11b}$$

where x_0 and z_0 are the x- and y-coordinates of the orthogonal projection of the ankle of the simplified model on the sole plane, $\hat{\theta}_1$ is the foot rotation angle relative to the cylinder obtained from the output of the balance controller, and θ_0 and θ_1 are the cylinder rolling angle and the foot rotation angle measured from the real robot, respectively. The determination of x_0, z_0, θ_0, and θ_1 will be discussed in the next section together with the measurement of other quantities.

To calculate the desired COP in the case of using the geta-feet model, I first compute the desired force and torque at the ankle joint of the simplified model that are required to realize the motion of the simplified model specified by the balance controller. The desired torque τ is determined by the feedback controller (2.5). The desired force can be calculated from the desired motion of the lump mass of the simplified model. From the state of the simplified model in the balance controller, the desired acceleration of the lump mass can be derived as

$$\ddot{x} = r_0\ddot{\theta}_0 + h(\ddot{\theta}_0 \cos\theta_0 - \dot{\theta}_0^2 \sin\theta_0) + l_0(\ddot{\theta}_{01} \cos\theta_{01} - \dot{\theta}_{01}^2 \sin\theta_{01}) \tag{2.12a}$$

$$\ddot{z} = -h(\ddot{\theta}_0 \sin\theta_0 + \dot{\theta}_0^2 \cos\theta_0) - l_0(\ddot{\theta}_{01} \sin\theta_{01} + \dot{\theta}_{01}^2 \cos\theta_{01}) \tag{2.12b}$$

where $\theta_{01} = \theta_0 + \theta_1$ and $\theta_{0,1}$, $\dot{\theta}_{0,1}$, and $\ddot{\theta}_{0,1}$ are obtained from the balance controller (2.6). Then, the desired force can be easily computed by

$$f_x = m_2\ddot{x} \tag{2.13a}$$

$$f_z = m_2(\ddot{z} + g). \tag{2.13b}$$

The desired COP can therefore be written as

$$x_{\mathrm{COP}} = x_0 + s\cos\hat{\theta}_0 \tag{2.14a}$$

$$z_{\mathrm{COP}} = z_0 - s\sin\hat{\theta}_0 \tag{2.14b}$$

Table 2.1. Common Sensors used on a Robot and Their Functions

Sensor	Function
inertial measurement unit (IMU)	global position and orientation of the floating base (hip)
joint angle sensors	joint angles
force/torque sensors	forces and moments applied by the environment to the ankles

where x_0 and z_0 give the orthogonal projection of the ankle in the simplified model on the sole plane, $\hat{\theta}_0$ is the rolling angle of the cylinder or the inclination angle of the foot specified by the balance controller, and s is the offset of the desired COP from the ankle projection in the sole plane, which can be calculated as

$$s = -\frac{(f_x \cos\hat{\theta}_0 - f_y \sin\hat{\theta}_0)h + \tau}{f_x \sin\hat{\theta}_0 + f_y \cos\hat{\theta}_0} \tag{2.15}$$

where h is the tooth height of the geta foot and τ is the desired ankle torque obtained from the the feedback controller (2.5) based on the simplified model.

2.5 System Measurement

A humanoid robot is usually equipped with multiple sensors for users to detect its current state. Common sensors include the inertial measurement unit (IMU), joint angle sensors, and force/torque sensors. The function of each type of sensors is described in Table 2.1. The IMU can be attached to any limb of the robot and gives the global position and orientation of that limb. In my case, the IMU is placed at the hip, which is taken to be the base of the robot. Assume that kinematic and dynamic models of the robot are available. Then, using those sensor data, many important information can be acquired, such as the actual COM and COP positions of the robot and the actual states of the simplified models, which are used as the input to the balance controller, as depicted in Fig. 2.3. Here I introduce how to determine those values based on the sensor data.

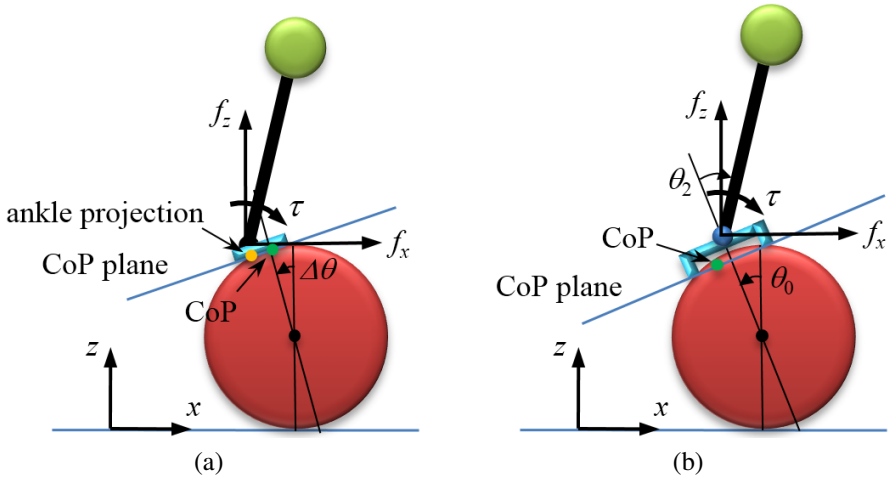


Figure 2.5. Measurement of the actual COP and the state of (a) the flat-feet model or (b) the geta-feet model.

2.5.1 COM Measurement

It is straightforward to measure the actual COM position of the robot. Based on the data from the IMU and joint angle sensors, the COM position can be easily calculated through forward kinematics using the available kinematic and dynamic models of the robot. By doing this, I also obtain the global position and orientation of each foot of the robot.

2.5.2 COP Measurement

The COP is a point in a plane passing through the contacts between the feet and the cylinder such that the net moment generated by the contact forces and moments about the point along any direction in the plane is zero, as depicted in Fig. 2.5. The plane is called the COP plane hereinafter. In the case that flat feet are used, the COP plane is chosen to be tangent to the cylinder passing through the contact edge (Fig. 2.5a). In the case that geta feet are used, the COP plane is the plane passing both contact edges at the toe and the heel

(Fig. 2.5b). The COP plane can be expressed as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \begin{bmatrix} \cos\alpha & 0 \\ 0 & 1 \\ -\sin\alpha & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.16}$$

where $[x_0 \quad y_0 \quad z_0]^T$ is an arbitrary point on the plane and α is the inclination angle of the plane, both of which can be determined from the positions and orientations of feet. To determine the point, the average position of two feet is used, while their average orientation is used to determine the inclination angle. The COP plane determined in this way tries to fit both feet, which may not perfectly align with each other on the cylinder.

From force/torque sensor data, I obtain the contact force and moment applied at each foot. Based on the global positions and orientations of feet, I can convert the forces and moments on both feet to the net force $\boldsymbol{f}$ and moment $\boldsymbol{n}$ applied to the robot with respect to the global frame. Then, the actual COP should satisfy the following equation

$$\begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \end{bmatrix} (\boldsymbol{n} + \hat{\boldsymbol{f}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}) = \mathbf{0} \tag{2.17}$$

where $\hat{\boldsymbol{f}}$ represents the cross product and $[x \quad y \quad z]^T$ is the actual COP position with respect to the global frame.

Substituting (2.16) into (2.17) yields

$$\boldsymbol{A} \begin{bmatrix} u \\ v \end{bmatrix} = \boldsymbol{b} \tag{2.18}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \end{bmatrix} \hat{\boldsymbol{f}} \begin{bmatrix} \cos\alpha & 0 \\ 0 & 1 \\ -\sin\alpha & 0 \end{bmatrix} \tag{2.19}$$

$$\boldsymbol{b} = -\begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \end{bmatrix} (\boldsymbol{n} + \hat{\boldsymbol{f}} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}). \tag{2.20}$$

Finally, solving (2.18) for u and v and substituting them into (2.16), I obtain the measured position of the actual COP in the global frame.

2.5.3 Configuration Measurement of the Simplified Models

In the frontal plane, the actual COM and COP positions are used directly as the input to the balance controller (2.10). As for the input to the balance controller in the sagittal plane, however, I need to measure the actual configuration of the simplified model from the robot, which is explained as follows.

For the flat-foot model, I first calculate the distance d from the orthogonal projection of the ankle on the COP plane to the actual COP in the sagittal plane, as shown in Fig. 2.5a. Then, the measured value of angle θ_1 is equal to d/r_0 and that of angle θ_0 is equal to $\alpha - d/r_0$, where r_0 is the radius of the cylinder. Finally, from the actual COM position relative to the ankle position, I can easily calculate the actual values of θ_2 and l.

The configuration measurement of the geta-foot model is the same as that of the flat-foot model except that the rolling angle θ_0 of the cylinder is taken directly to be the inclination angle of robot feet.

Again, it should be noted that all the quantities discussed herein and hereinbefore are the average over both feet of the robot when the robot is in the double-support phase. Table 2.2 summarizes their measurement.

Table 2.2. Measurement of Quantities

Quantity	Required Sensor Data
robot feet positions/orientations	forward kinematics based on the IMU and joint sensor data
actual COM position	forward kinematics based on the IMU and joint sensor data
actual COP position	force/torque sensor data and robot feet positions/orientations
inclination angle of the COP plane	robot feet orientations
ankle position of the simplified model	robot feet positions/orientations
ankle projection on the COP plane	robot feet positions/orientations
rolling angle θ_0	robot feet positions/orientations and actual COP position (flat-feet model) robot feet orientations (geta-feet model)
foot rotation angle θ_1	robot feet positions/orientations and actual COP position (only for the flat-feet model)
ankle joint angle θ_2	ankle position of the simplified model, actual COM position, and inclination angle of the COP plane
length change l	ankle and COM positions

2.6 Full-Body Mapping and Simulation

The desired COP obtained from the balance controllers only tells the high-level information for maintaining the balance of the robot on the cylinder. To actually control the robot, full-body control commands need to be determined. Assume that the robot in simulation uses torque control, which means that the robot takes joint torques as the actual command to activate and control its motion. Here I discuss how to compute the joint torques for controlling the robot to follow the desired COP and maintain balance on the cylinder, followed by simulation results.

2.6.1 Torque Mapping

I calculate the joint torques using the method proposed by Yamane and Hodgins (2009). Originally, it considers only the motion of the robot itself. In order to actively control

cylinder's motion, I extend the method to including the cylinder with the robot and consider their dynamics all together.

The dynamics of the whole system, including the robot and the cylinder, can be described by the following equation

$$\begin{bmatrix} \boldsymbol{M}_r & \mathbf{0} \\ \mathbf{0} & \boldsymbol{M}_c \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_r \\ \ddot{\boldsymbol{q}}_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_r \\ \boldsymbol{c}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{N}^T \boldsymbol{\tau} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_r^T & \mathbf{0} \\ \boldsymbol{J}_{c1}^T & \boldsymbol{J}_{c2}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix}. \tag{2.21}$$

where $\boldsymbol{M}_r$ and $\boldsymbol{M}_c$ are the joint-space inertia matrices of the robot and the cylinder, respectively, $\boldsymbol{c}_r$ and $\boldsymbol{c}_c$ are the sums of Coriolis, centrifugal and gravity forces for them, $\boldsymbol{\tau}$ comprises the joint torques and $\boldsymbol{N}$ is the matrix used to map the joint torques into the generalized forces, $\boldsymbol{f}_1$ comprises the contact forces/moments between the robot and the cylinder and $\boldsymbol{J}_r$ and $\boldsymbol{J}_{c1}$ are the Jacobian matrices whose transposes convert $\boldsymbol{f}_1$ into the generalized forces for the robot and the cylinder, respectively, and $\boldsymbol{f}_2$ comprises the contact force/moment between the cylinder and the ground and $\boldsymbol{J}_{c2}$ is the Jacobian matrix whose transpose maps $\boldsymbol{f}_2$ into the generalized forces for the cylinder.

The goal of the method (Yamane and Hodgins, 2009) is to compute the joint torques for the robot to realize the desired COP as well as the desired joint accelerations $\hat{\ddot{q}}$ and the desired foot accelerations $\hat{\ddot{r}}$ for achieving a given reference motion. In the case herein, the reference motion for the robot consists merely of a single full-body standing pose on the cylinder. The desired accelerations of joints, including the joints of both the robot and the cylinder, are determined by the traditional proportional-derivative control law as

$$\hat{\ddot{q}} = \ddot{q}_{ref} + k_d(\dot{q}_{ref} - \dot{q}) + k_p(q_{ref} - q) \tag{2.22}$$

where q and $\dot{q}$ are the current joint angle and velocity, q_{ref}, $\dot{q}_{ref}$, and $\ddot{q}_{ref}$ are the reference joint angle, velocity, and acceleration, and k_p and k_d are proportional and derivative gains.

The desired accelerations $\hat{\dot{r}}$ of robot's feet, consisting of linear and angular components, are determined using the same control law as (2.22).

However, tracking the desired COP and realizing the reference pose may conflict with each other. Hence, an optimization problem is formulated to compute the joint torques that respect both of them. Free variables of the optimization problem include joint torques $\boldsymbol{\tau}$ and contact forces $\boldsymbol{f}$, while joint accelerations $\ddot{\boldsymbol{q}}$ depend completely on $\boldsymbol{\tau}$ and $\boldsymbol{f}$ according to the full-body dynamics equation (2.21). The cost function to be minimized consists of several terms addressed as below.

The COP error, namely the error in tracking the desired COP obtained from the balance controller, can be expressed as the magnitude of the resultant moment around the desired COP as

$$Z_{COP} = \frac{1}{2}\boldsymbol{f}_1^T \boldsymbol{P}^T \boldsymbol{W}_P \boldsymbol{P} \boldsymbol{f}_1 \tag{2.23}$$

where $\boldsymbol{P}$ is the matrix that maps $\boldsymbol{f}_1$ to the resultant moment around the desired COP.

The error Z_q from the desired joint accelerations can be written as

$$Z_q = \frac{1}{2}(\hat{\ddot{\boldsymbol{q}}} - \ddot{\boldsymbol{q}})^T \boldsymbol{W}_q (\hat{\ddot{\boldsymbol{q}}} - \ddot{\boldsymbol{q}}). \tag{2.24}$$

From (2.21) it follows that the joint accelerations $\ddot{\boldsymbol{q}}$ can be expressed as a linear function of $\boldsymbol{\tau}$ and $\boldsymbol{f}$.

The error Z_τ from the desired feet accelerations can be written as

$$Z_c = \frac{1}{2}(\hat{\ddot{\boldsymbol{r}}} - \ddot{\boldsymbol{r}})^T \boldsymbol{W}_c (\hat{\ddot{\boldsymbol{r}}} - \ddot{\boldsymbol{r}}) \tag{2.25}$$

where $\hat{\ddot{\boldsymbol{r}}}$ is the desired feet accelerations and can be determined using the same control law as (2.22). The relationship between the generalized velocity $\dot{\boldsymbol{q}}$ and the velocity of one foot $\dot{\boldsymbol{r}}_i$ is given by

$$\dot{\boldsymbol{r}} = \boldsymbol{J}_r \dot{\boldsymbol{q}}_r. \tag{2.26}$$

Differentiating (2.26) yields the foot acceleration:

$$\ddot{\boldsymbol{r}} = \boldsymbol{J}_r \ddot{\boldsymbol{q}}_r + \dot{\boldsymbol{J}} \dot{\boldsymbol{q}}_r. \tag{2.27}$$

From (2.21) and (2.27), $\ddot{\boldsymbol{r}}$ is completely determined by $\boldsymbol{\tau}$ and $\boldsymbol{f}$.

Other terms may include the magnitudes of joint torques and contact forces:

$$Z_m = \frac{1}{2}\boldsymbol{\tau}^T \boldsymbol{W}_\tau \boldsymbol{\tau} + \frac{1}{2}\boldsymbol{f}^T \boldsymbol{W}_f \boldsymbol{f}. \tag{2.28}$$

Combining all the terms, the cost function can be written in a quadratic form

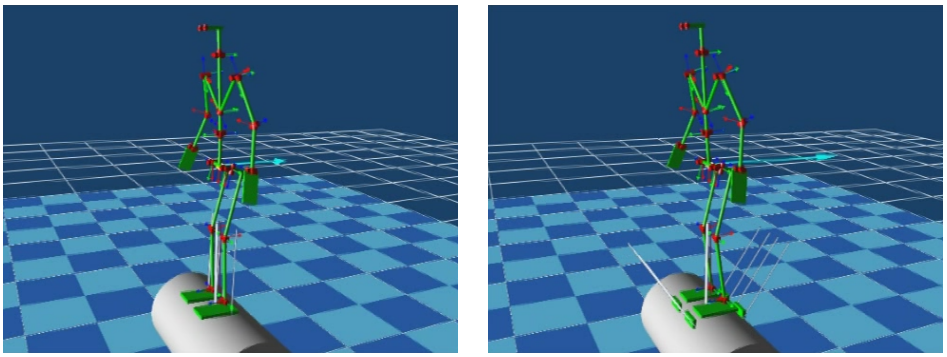
$$Z = \frac{1}{2}\boldsymbol{y}^T \boldsymbol{A} \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{b} + \boldsymbol{c} \tag{2.29}$$

where $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{\tau}^T & \boldsymbol{f}^T \end{bmatrix}^T$. The analytic solution to the optimization problem (2.29) is $\boldsymbol{y} = -\boldsymbol{A}^{-1}\boldsymbol{b}$. Then the resulting optimized joint torques $\boldsymbol{\tau}$ are sent to the robot as depicted in Fig. 2.3.

2.6.2 Simulation

I use the dynamics simulator with rigid-contact model developed by the University of Tokyo (Yamane and Nakamura, 2008a,b) to conduct my simulation. The simulator computes the actual contact forces and joint accelerations after applying the optimized joint torques obtained as above to the robot, considering the complementary constraint on the contact force and the contact link motion. This problem can be formulated as a linear complementary problem and solved by the algorithm developed by Yamane and Nakamura (2008b). This is out of the scope of this dissertation and readers are referred to the relevant publications.

The humanoid robot model used in the simulations has 25 joints and 31 DOFs including the translation and rotation of the floating base. Each leg has 7 joints (pitch, roll, yaw at both the hip and the ankle and pitch at the knee). I consider only 4 joints in each arm (pitch,



(a) (b)

Figure 2.6. Simulation of a biped robot with (a) flat feet or (b) geta feet on a rolling cylinder. The blue arrow represents the external disturbance applied to the robot.

roll, yaw at the shoulder and pitch at the elbow) and fix wrist joints. There are 3 joints in the torso. The robot model is about 1.7 meters tall and 65 kg in weight. The radius of the cylinder is 0.254 m.

Both flat and geta feet are tested in simulation, as shown in Fig. 2.6. A disturbing force is applied at the hip of robot in the sagittal plane to verify the effectiveness of the balance controller. In the case of flat feet, the robot can successfully balance on the cylinder under a disturbing force of 20 N applied for 100 ms. It is noticed in simulation that it is hard for the robot to completely stop on top of the cylinder and immobilize the cylinder. In the case of geta feet, by contrast, the robot can survive under a bigger disturbance, a 40 N force applied for 100 ms, and freeze on the cylinder. This is due to that the robot has a much bigger support region and the model has one less DOF by using this special design of feet. A video of the simulation can be found on `http://www.cs.unc.edu/~yuzheng/dissertation/`.

2.7 Experiments on the Sarcos Humanoid Robot

Fig. 2.7 shows the experimental setup on the Sarcos humanoid robot for testing the balance controller. In the hardware experiment, I cannot use the torque control technique as I do in simulation because there is no accurate dynamics model for this real robot, which is

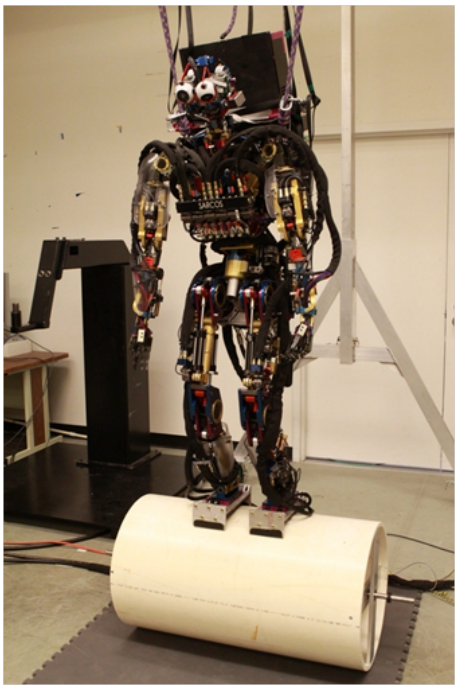
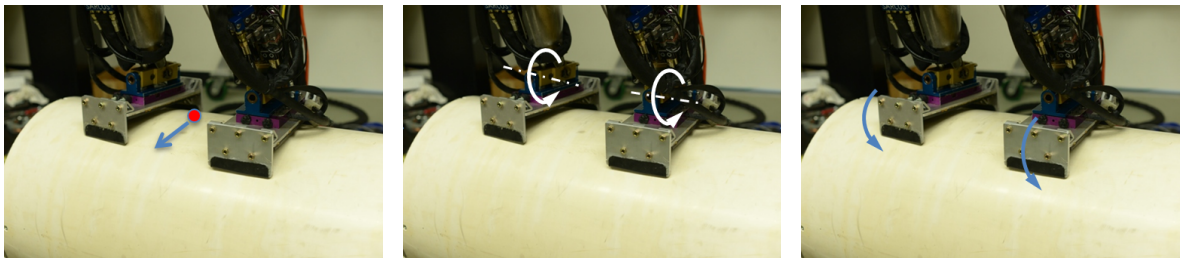


Figure 2.7. Experimental setup on the Sarcos humanoid robot.

required in computing the joint torques. Hence, I switch to the position control and compute the desired joint angles for the robot in order to maintain balance. Moreover, from the simulation results it has been seen that it is easier for the robot to stand on the cylinder using geta feet. Thus, I use geta feet rather than flat feet in the experiment. By doing this, it also becomes easier for the robot to reach an initial pose that is close to the static equilibrium.

The key idea of the position mapping is to convert the desired COP change to the desired joint angles for the robot. Fig. 2.8 illustrates this conversion. Suppose that the desired COP needs to move forward in comparison with the current COP, as depicted in Fig. 2.8a. This is equivalent to increasing the ankle torque (Fig. 2.8b) or to extending the feet (Fig. 2.8c). Therefore, I can convert the desired COP to the change of the foot orientation by the following law:

$$u = (x^d_{\mathrm{COP}} - x^0_{\mathrm{COP}}) + K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \tag{2.30}$$



(a) (b) (c)

Figure 2.8. Conversion of the desired COP to the foot rotation. (a) The desired COP moving forward. (b) Increasing the ankle torque in the direction as shown. (c) Extending the feet.

where x^d_{COP} and x^0_{COP} are the desired and initial positions of the COP in the sagittal plane, respectively, $e(t)$ is the error between the desired COP and the actual COP, and K_p, K_i, and K_d are the proportional, integral, and derivative gains. Finally, I take u/R as the desired rotation of feet from their initial orientation, where R is a value that needs to be tuned in the experiment. With proper settings of K_p, K_i, K_d, and R, the Sarcos humanoid robot can balance on the cylinder. A video showing the experiment is provided on `http://www.cs.unc.edu/~yuzheng/dissertation/`.

2.8 Conclusions and Future Work

This chapter discusses the balance control of a humanoid robot standing on a dynamic object, a free-rolling cylinder in this particular piece of work. The control framework comprises a balance controller and a full-body mapping unit. The balance controller is designed based on a simplified model and produces the desired COP/COM positions for the robot to maintain balance, while the full-body mapping unit computes the control command for the robot, namely joint torques or angles, to achieve those positions and realize a desired full-body pose. To enable the robot actively manipulate cylinder's motion rather than passively adapt to it, the dynamics of the cylinder is considered in the balance controller and the full-body mapping unit. It is shown by simulation that a torque-controlled robot can successfully recover balance from disturbances on a cylinder by using this control

technique. I also conduct hardware experiments on a position-controlled robot and use a joint angle mapping instead of joint torque optimization. The robot can balance on a free-rolling cylinder for at least 20 s.

This control method has the limitation that the dimension and the dynamic parameters of the object need be known. In some situations, there may be no opportunity to acquire those parameters beforehand. It is a valuable direction to extend the current work to handling unknown objects. In addition, one of the problems with robot control using joint torques is that accurate kinematic and dynamic models of the robot are required; otherwise, the computed joint torques will not produce the desired motion on the robot. It is usually hard to obtain accurate models of a humanoid robot, which has so many DOFs, and modelling errors are inevitable. That is why I can only use position control in the hardware experiment, though the robot can potentially use torque control. How to deal with modelling errors in the balance control of a torque-controlled humanoid robot is also an important problem that deserves further exploration.

CHAPTER 3: MANIPULATING A DYNAMIC OBJECT BY ACTIVE WALKING

3.1 Introduction

Only being able to balance on the cylinder will get the robot nowhere. In order to enhance its mobility, the robot need be able to actively move the cylinder in some way. Based on the balance controller presented in the previous chapter, this chapter discusses how the robot can actively manipulate the rolling of the cylinder through biped walking on top of it, which is challenging even for normal humans.

3.1.1 Main Results

In this chapter, I present two methods for generating biped walking gaits for a humanoid robot to walk on and roll the cylinder. In the first method, I allow the walking gait to have a double-support phase, in which both robot feet are in contact with the cylinder. The rolling of the cylinder happens only during the double-support phase, whereas in the single-support phase the robot stands still on one foot and takes a step on the cylinder. Hence, the stepping motion of the swing foot and the rotation of the cylinder happen in sequence. Since there is a pause between steps and the cylinder does not roll continuously, this method generates an intermittent walking behavior. Also, having the double-support phase, the COP can gradually shift from the left side to the right or vice versa, which does not require a fast COM motion and makes the walking gait more static and easier for the robot to realize.

In the second method, I consider a more dynamic walking gait, which does not involve a double-support phase. During walking, the robot is supported by only one foot all the time. While the swing foot touches down, the supporting foot lifts up immediately. The swinging of the foot and the rolling of the cylinder happen simultaneously and the cylinder

keeps rolling without a stop between steps. Thus, the generated walking behavior is more continuous. Furthermore, I expect the walking gait to be identical between steps in terms of the states of the robot and the cylinder, and such a gait is called a cyclic walking gait.

3.1.2 Organization

This chapter is organized as follows. Section 3.2 summarizes the previous work on biped walking generation. Sections 3.3 and 3.4 discuss the generating of the static walking gait and the cyclic walking gait, respectively. Conclusions and future work are given in Section 3.5.

3.2 Previous Work

3.2.1 Biped Locomotion Generation

In generating the walking pattern for a biped robot, it is often assumed that the footsteps are given, and the problem becomes how to compute the joint angles such that the resulting motion satisfies the balance condition and the given footsteps. Based on the given footsteps, one may first determine a ZMP or COP trajectory and compute a physically consistent COM trajectory of the robot using a simplified model, such as an inverted pendulum (Kajita and Tanie, 1995; Kajita et al., 2001). Then, the joint angles of the robot can be calculated using inverse kinematics, according to the COM trajectory and the footsteps. This approach has been successfully used to generate biped walking patterns in a stationary environment (Sugihara et al., 2002; Kajita et al., 2003). In an environment with known obstacles, several algorithms have been proposed to compute footsteps and a collision-free path for a biped robot (Kuffner et al., 2001; Chestnutt et al., 2005; Ayaz et al., 2009) such that the walking pattern can be generated later in this way. However, motions generated by this approach usually do not look human-like, though some work tried to mimic human walking motion by adding single toe support phase and characterized swing leg motions (Miura et al., 2011). Another issue is that the ZMP feasibility criterion is only applicable to horizontal terrains.

Hirukawa et al. (2006) used a more general criterion based on feasible contact forces in the design of a walking pattern generator.

Humanoid locomotion can also be generated in a reverse way. One may first have a reference COM or upper-body motion for a humanoid robot, which can be acquired from a captured human motion or a motion planner. Then, based on a simplified dynamics model, the required COP trajectory can be quickly computed from the reference motion. From the COP trajectory, a sequence of appropriate footsteps can be determined to cover it. Finally, the full-body trajectories in terms of joint angles for the robot can be calculated through inverse kinematics based on the COM or upper body motion and the sequence of footsteps. Nevertheless, there is not much work on this kind of approaches (Sugihara, 2008).

3.2.2 Limit Cycle Walking of Passive Walkers

Limit cycle walking is a fundamental topic in the research of passive biped robots. McGeer (1990) first demonstrated that a passive biped robot can walk down a slope in a steady periodic gait without any active control. The only energy supply to the robot is the potential energy, which compensates the loss of energy when the swing leg hits the ground. After McGeer's pioneering work, many researchers investigated passive biped walking. Goswami et al. (1998) and Garcia et al. (1998) verified the existence and the stability of limit cycles. Osuka and Kirihara (2000) first demonstrated this symmetric motion on a real passive robot. Collins et al. (2001) built the first three-dimensional passive biped robot with knees. Ikemata et al. (2003; 2008) studied several factors that may affect the stability of a limit cycle, such as the support exchange, the stabilization of a fixed point, and the motion of the swing leg. Freidovich et al. (2009) proposed a faster way to seek both stable and unstable limit cycles than traditional numerical routines.

Without any actuation or energy input, a passive walking robot can only walk on a declining slope. However, with the help of one or more actuators to compensate the energy loss at heel strike, powered passive walkers are able to walk on flat, level, or uphill

ground and have higher capability to handle disturbances. One energy-efficient way to add actuation is the use of actuated ankles (Kuo, 2002; Collins et al., 2005; Hobbelen and Wisse, 2008a,b; Franken et al., 2008). Using the ankle push-off not only decreases the energy use (Kuo, 2002; Collins et al., 2005) but also increases limit cycle walkers' ability to reject disturbances (Hobbelen and Wisse, 2008a). The use of ankle actuation also allows a robot to achieve different walking speed in limit cycle walking (Hobbelen and Wisse, 2008b). It has also been shown with simulation that pushing off before the swing leg hits the ground is energetically more efficient than pushing off after the heel strikes (Franken et al., 2008). Actuation can also be added at the hip joint (Dertien, 2006). Harada et al. (2010) applied the limit cycle based walking generation to a model of present humanoid robots with more active joints and flat feet.

3.3 Static Walking Gait Generation

In this section, I discuss how to generate a walking gait simply based on the balance controller presented in the previous chapter. It is expected that, by using the balance controller, the robot will roll the cylinder to recover its feet to be horizontal once the feet lean on top of the cylinder. Then, I use this property to design a walking gait generator and enable the robot to intensionally roll the cylinder by changing foot locations on top of it. Fig. 3.1 depicts a framework for the walking gait generator, which consists primarily of planners for frontal and sagittal motions and the balance controller. The frontal and sagittal motion planners generate the trajectories of the COM and the feet so that the full-body reference motion, expressed as joint trajectories, can be calculated through inverse kinematics.

3.3.1 Frontal Motion Planner

The goal of the frontal motion planner is to generate a feasible COM trajectory in the frontal plane for the robot to follow such that the actual COP can move to the beneath of

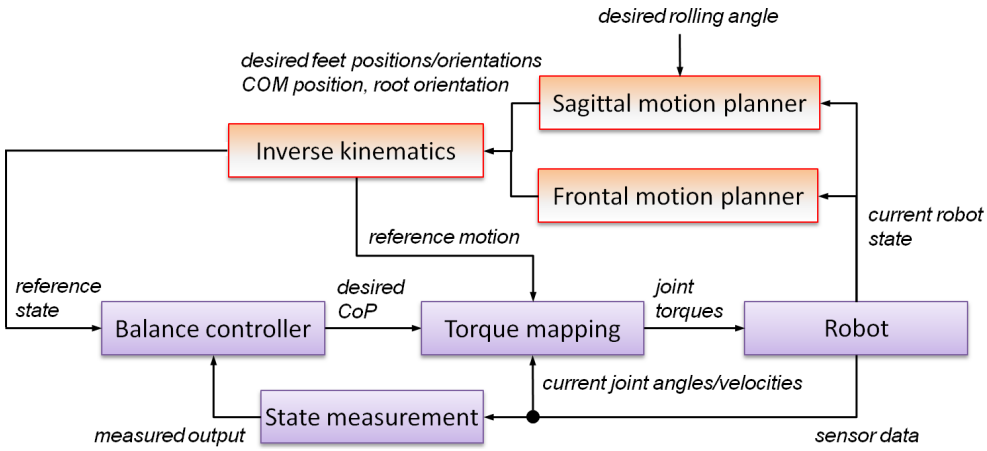


Figure 3.1. Framework for walking motion generation on a rolling cylinder.

the supporting foot and the swing foot can lift up. Yamane and Hodgins (2010) proposed a method to modify the COM trajectory of a reference motion such that the corresponding desired COP can stay within the support region. I use this method to generate the reference COM trajectory in the frontal plane. Following the method (Yamane and Hodgins, 2010), I first discretize the state-space equation of the balance controller (2.10) and obtain

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k \tag{3.1}$$

where $\boldsymbol{x}_k$ is the state and u_k is the input at sampling time k. In contrast to the balance controller (2.10), here I do not include the observer in (3.1) because I am planning the motion and do not have real measurements. As a result, u_k comprises only a COM position. The output is chosen to be the COP position and can be written as

$$y_k = \boldsymbol{C}\boldsymbol{x}_k \tag{3.2}$$

where $\boldsymbol{C} = [1 \quad 0 \quad 0 \quad 0]^T$.

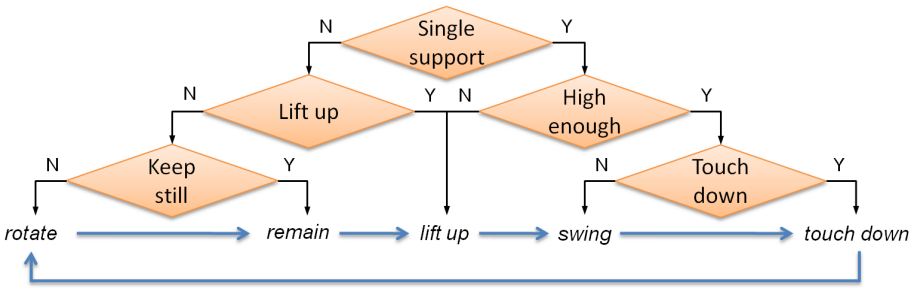


Figure 3.2. Framework for sagittal walking motion generation on a rolling cylinder.

Given an initial state $\boldsymbol{x}_0$ and the COM positions for the next n frames, u_k ($k = 0, 1, \ldots, n-1$), through (3.1) I can predict the COP position in n frames by

$$y_n = \boldsymbol{C}\left(\boldsymbol{A}^n \boldsymbol{x}_0 + \boldsymbol{M}\boldsymbol{u}\right) \tag{3.3}$$

where

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A}^{n-1}\boldsymbol{B} & \boldsymbol{A}^{n-2}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} u_0 & u_1 & \cdots u_{n-1} \end{bmatrix}^T. \tag{3.4}$$

Suppose that y_{ref} is the desired COP position in n frames. Then, I can compute a reference COM trajectory $\boldsymbol{u}$ for the n frames such that y_n approaches y_{ref} by minimizing the following cost function:

$$Z_d = \frac{1}{2}(y_{\text{ref}} - y_n)^2 + \frac{1}{2}\left(\hat{\boldsymbol{u}} - \boldsymbol{u}\right)^T \boldsymbol{W}\left(\hat{\boldsymbol{u}} - \boldsymbol{u}\right) \tag{3.5}$$

where $\hat{\boldsymbol{u}}$ is a nominal COM trajectory and can be simply taken to be $\hat{u}_k = x_0 + k(y_{\text{res}} - x_0)/n$. The purpose of the first term of Z_d is to bring the COP as close as possible to the desired value y_{ref}, while that of the second term is to prevent the generated COM trajectory deviating from a reasonable region.

3.3.2 Sagittal Motion Planner

The sagittal motion planner generates the desired trajectories of the feet and the COM of the robot in the sagittal plane. Fig. 3.2 shows a flowchart for foot trajectory generation.

The robot undergoes the double support phase and the single support phase alternately on the cylinder. The foot to be lifted up and put down at a different location is called the swing foot, while the other foot that supports the robot is called the supporting foot. The swing foot will have one of the following five actual/desired states:

a) keeping contact with the cylinder and rotating together with the supporting foot to roll the cylinder;

b) remaining at its current position and orientation while the COP moves to the side of the supporting foot;

c) lifting up while the supporting foot keeps still on the cylinder;

d) swinging to the target position for touching down;

e) touching down at the target position on the cylinder.

The swing foot needs to follow the desired states in the sequence as shown in Fig. 3.2 in order to accomplish a walking step. After the swing foot lands on the cylinder, it becomes the new supporting foot and the supporting foot becomes the new swing foot for the next step. For different desired states, corresponding reference motions are generated and used in computing joint torques as discussed in Section 2.6.1. Note that the actual state of a foot may not immediately match the desired one. In what follows, I discuss the generation of reference motion for each desired state and the condition for triggering a desired state.

First, I determine if a foot is in support by checking the contact force between it and the cylinder. The contact force is measured by the force/torque sensor attached to the foot. If the contact force is bigger than a certain threshold, then the foot is considered to be in contact with the cylinder and support the robot. In the case that both feet are in support, I continue to determine whether the swing foot should rotate or lift up or remain in support. Since the desired states happen in a fixed sequence, the state that is expected to happen next

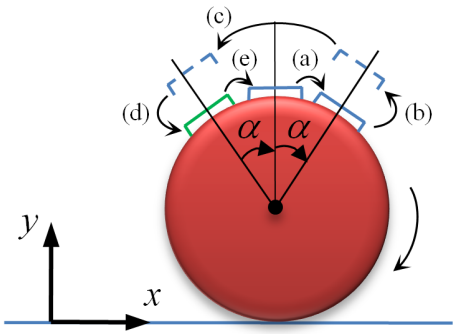


Figure 3.3. Illustration of the motion of the swing foot in a step. The swing foot (in the blue color) (a) rotates together with the supporting foot (not shown) and rolls the cylinder for angle α, (b) lifts up, (c) moves backwards, and then (d) touches down. Once touching down, the swing foot becomes the supporting foot (in the green color) for the next step and (e) rotates and rolls the cylinder for another angle α. (a) and (e) happen at the same time in one step on the swing foot and the supporting foot, respectively.

can be known directly from the state that has just happened, and it is only required to check if the corresponding condition is triggered.

Fig. 3.3 illustrates the motion of the swing foot in one step. Assume that the cylinder is required to roll forward for an angle α in every step. At the beginning of a step, the swing foot, namely the supporting foot in the previous step, should be on top of the cylinder, while the supporting foot, namely the swing foot after touching down at the end of the previous step, is behind the top. To make the supporting and swing feet rotate about the cylinder, which as a result causes the cylinder to roll, I first measure the state of the simplified model based only on the position and orientation of the supporting foot and use the measured state as the input to the balance controller. Since the supporting foot is behind the top of the cylinder, the balance controller should be able to bring it to the top, as shown by arrow (e) in Fig. 3.3, which causes the cylinder to roll forward due to the friction between the foot and the cylinder. The swing foot can rotate simultaneously with the supporting foot and the cylinder as long as it remains in contact with the cylinder.

To realize this motion, I also change the reference pose in the torque mapping as follows. I choose the target position of the supporting foot to be on top of the cylinder and its target orientation to be horizontal. The target position and orientation of the swing foot can be

obtained by rotating it from the top of the cylinder for the angle α about the axis of the cylinder. The COM of the robot is chosen above the center of the cylinder, while the orientation of the root of the robot keeps original. Then, all joint angles of the robot can be calculated through inverse kinematics.

After the cylinder rotates for the angle α and both feet reach the aforementioned target positions, the swing foot may not be able to lift up immediately, since it may still support the robot. It should remain in contact with the cylinder until the actual COP is completely under the supporting foot, which implies that the entire weight of the robot is sustained by the supporting foot. In order to reduce the actual contact force expected on the swing foot and help move the COP to the supporting foot side, I increase the weights $\boldsymbol{W}_f$ in the cost function term (2.28) of joint torque optimization that correspond to the contact forces on the swing foot. Once the COP is under the supporting foot, I set the target position for the swing foot to be above its current position, as depicted in Fig. 3.3, and the torque mapping unit should compute the required joint torques for lifting it up to this target position.

The planning of swing trajectory is straightforward. The location for touching down is obtained by rotating the supporting foot backward for the angle α. During the lifting-up, swinging, and touching-down of the swing foot, the supporting foot should remain on top of the cylinder. Once the swing foot makes a stable contact with the cylinder, it becomes the supporting foot for the next step and the previous supporting foot becomes the swing foot, and they follow the same sequence as just described.

3.3.3 Simulation Results

Fig. 3.4 shows the generated walking behavior using the proposed method in full-body dynamics simulation, which uses the same simulator and parameters as in Section 2.6.2. In every step, the robot is required to roll the cylinder for 0.1 rad. The reference and actual positions of the COM and the COP of the robot during the first six steps are plotted in Fig. 3.5. It can be seen that the actual COM and COP positions track the desired values very well,

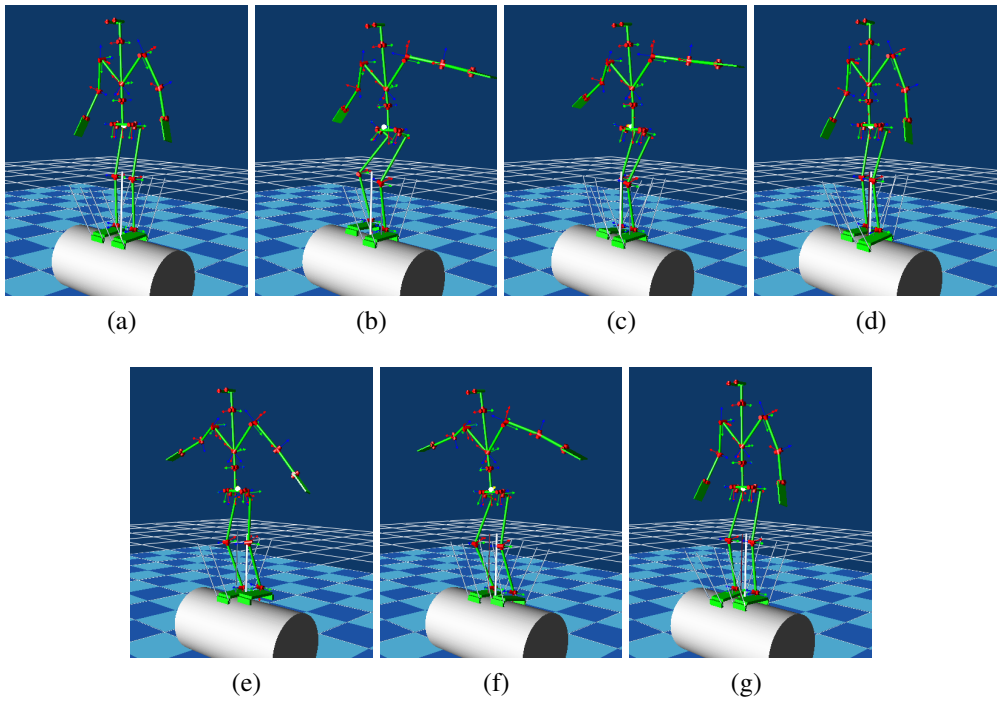


Figure 3.4. Snapshots of biped walking on a rolling cylinder. (a) Initial pose. (b) Moving the weight to the left foot and lifting up the right foot. (c) Touching down the right foot at a position behind the left foot. (d) Rotating the cylinder forward while moving the weight to the right. (e)-(g) Repeating this procedure with the right foot as the supporting foot and the left foot as the swing foot.

which implies that the robot realizes the generated reference walking motion. Furthermore, based on the change of the COM position in the sagittal plane from the beginning of the first step to the end of sixth step (Fig. 3.5a), it can be confirmed that the cylinder reaches the desired rolling angle for every step. A video of this simulated walking behavior is shown on `http://www.cs.unc.edu/~yuzheng/dissertation/`.

3.4 Cyclic Walking Gait Generation

The walking motion generated using the aforementioned method involves a double-support phase such that the COM of the robot is always above the support region. As a consequence, the walking motion is safe but static. In this section, I focus on the sagittal motion and investigate how to achieve a more dynamic walking behavior in a cyclic pattern

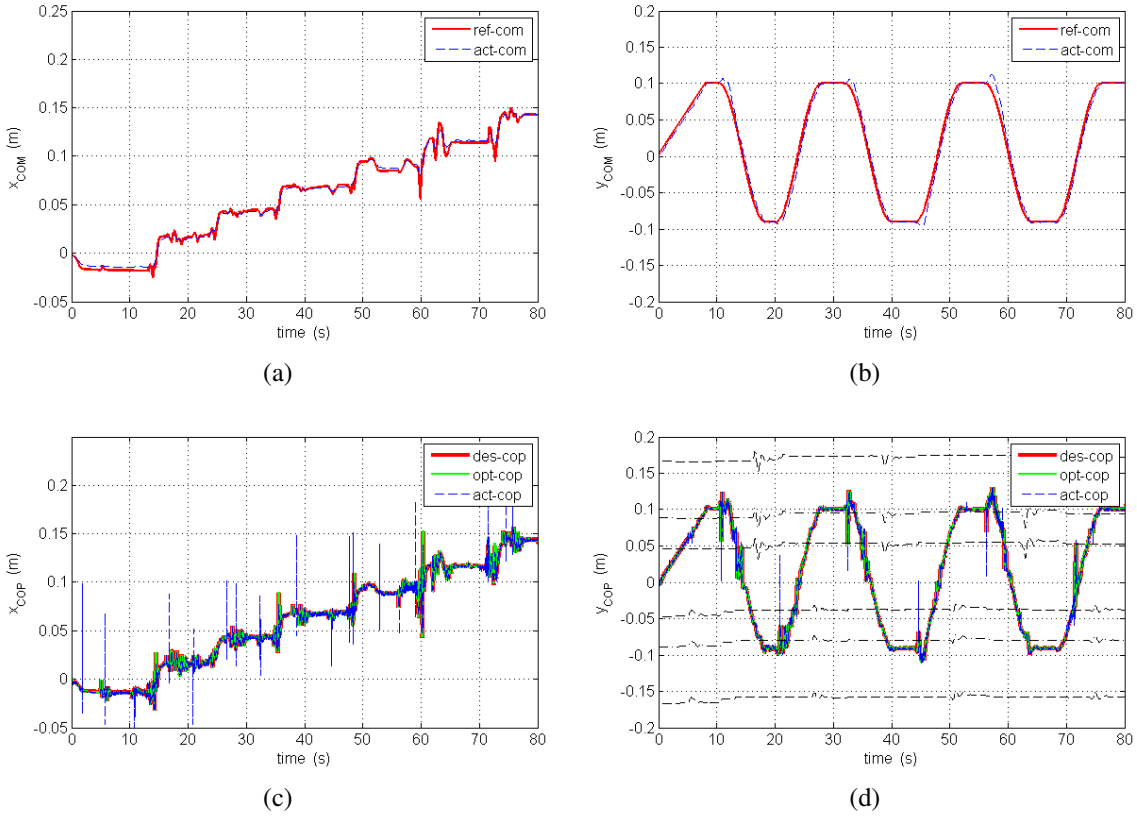


Figure 3.5. Graphs of (a,b) the reference and actual COM trajectories and (c,d) the desired, optimized, and actual COP trajectories.

without having a double-support phase and to roll the cylinder continuously at a desired speed. Therefore, it is assumed in the following discussion that the supporting foot lifts up immediately once the swing foot touches down.

3.4.1 Equation of Motion

For the purpose of walking gait planning, I consider the sagittal-plane simplified model given in Section 2.4.1 with the feedback controller given in Section 2.4.4 but do not include the observer. Since the model consists of the cylinder, a foot representing the supporting foot of the robot, and a lump mass representing the robot's COM, from the motion of the simplified model I can obtain the motion of the counterparts in the real system consisting of

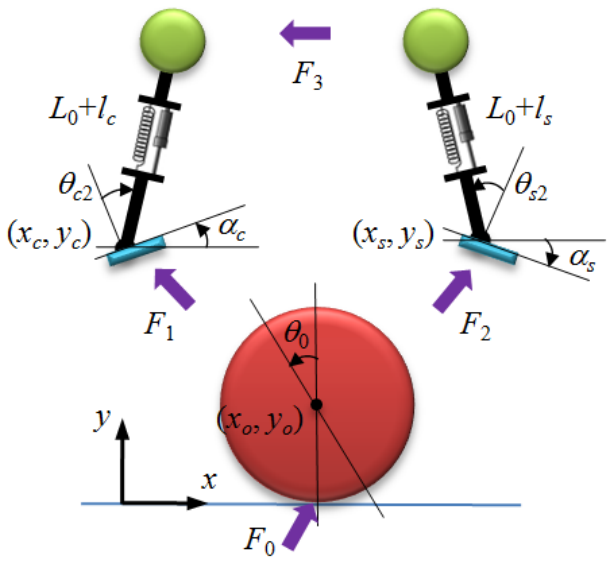


Figure 3.6. Collision model. The motion of the cylinder is described using variables x_o, y_o, θ_0, while that of each leg is described using variables $x, y, \alpha, \theta_2, l$. The subscripts c and s represent the colliding (swing) leg and the supporting leg, respectively.

the cylinder and a real robot, and the full-body motion of the robot can be calculated via inverse kinematics.

I derive the equation of motion for the simplified model with the feedback controller as follows. Substituting (2.5) into (2.3), I first obtain

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{BK})\boldsymbol{x}. \tag{3.6}$$

Then, the solution to the differential equation (3.6) can be written as

$$\boldsymbol{x} = e^{(\boldsymbol{A}-\boldsymbol{BK})t}\boldsymbol{x}_0 \tag{3.7}$$

where $\boldsymbol{x}_0$ is the initial state. From (3.7) it can be seen that the motion of the simplified model is completely determined by its initial state $\boldsymbol{x}_0$.

3.4.2 Collision Model

The motion of the robot with the cylinder during the single-support phase can be described by (3.7). While the swing foot touches and the supporting foot leaves the cylinder, the induced impact may cause a sudden change in the state of the whole system. To model this effect, I present a collision model, which describes the motions of the cylinder and two legs separately using the variables summarized in Fig. 3.6. The configuration of the cylinder is represented by (x_o, y_o) and θ_0, which indicate the position and orientation of the cylinder, respectively. The configuration of a leg is determined by another five parameters; that is, (x, y) to represent the position of the ankle joint, α the angle of the foot with respect to the horizontal plane, and θ_2 and l have the same meaning as in the simplified dynamics model. Therefore, I describe the configurations of the cylinder, the swing leg, and the supporting leg respectively by the following vectors:

$$\boldsymbol{q}_o = [x_o \quad y_o \quad \theta_0]^T \tag{3.8}$$

$$\boldsymbol{q}_c = [x_c \quad y_c \quad \alpha_c \quad \theta_{c2} \quad l_c]^T \tag{3.9}$$

$$\boldsymbol{q}_s = [x_s \quad y_s \quad \alpha_s \quad \theta_{s2} \quad l_s]^T. \tag{3.10}$$

The subscripts “o”, “c”, and “s” represent the cylinder, the swing (colliding) leg, and the supporting leg, respectively.

The position of the contact between the cylinder and the floor can be written in terms of $\boldsymbol{q}_o$ as

$$\boldsymbol{p}_F = \begin{bmatrix} x_o - r_0\theta_0 \\ y_o - r_0 \end{bmatrix}. \tag{3.11}$$

Then the Jacobian matrix of $\boldsymbol{p}_F$ with respect to $\boldsymbol{q}_o$ is

$$\boldsymbol{J}_{FO} = \frac{\partial \boldsymbol{p}_F}{\partial \boldsymbol{q}_o} = \begin{bmatrix} 1 & 0 & -r_0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{3.12}$$

The position of the contact between the cylinder and the swing leg on the cylinder can be expressed as

$$\boldsymbol{p}_{CO} = \begin{bmatrix} x_o + r_0\theta_0 \cos\alpha_c \\ y_o - r_0\theta_0 \sin\alpha_c \end{bmatrix}. \tag{3.13}$$

Then the Jacobian matrix of $\boldsymbol{p}_{CO}$ with respect to $\boldsymbol{q}_o$ is

$$\boldsymbol{J}_{CO} = \frac{\partial \boldsymbol{p}_{CO}}{\partial \boldsymbol{q}_o} = \begin{bmatrix} 1 & 0 & r_0 \cos\alpha_c \\ 0 & 1 & -r_0 \sin\alpha_c \end{bmatrix}. \tag{3.14}$$

The position of the contact between the swing leg and the cylinder on the leg can be expressed in terms of $\boldsymbol{q}_c$ as

$$\boldsymbol{p}_{OC} = \begin{bmatrix} x_c - \lambda \cos\alpha_c \\ y_c + \lambda \sin\alpha_c \end{bmatrix} \tag{3.15}$$

where $\lambda = (x_c - x_o)\cos\alpha_c - (y_c - y_o)\sin\alpha_c$ is treated as a constant. Then the Jacobian matrix of $\boldsymbol{p}_{OC}$ with respect to $\boldsymbol{q}_c$ can be calculated by

$$\boldsymbol{J}_{OC} = \frac{\partial \boldsymbol{p}_{OC}}{\partial \boldsymbol{q}_c} = \begin{bmatrix} 1 & 0 & \lambda \sin\alpha_c & 0 & 0 \\ 0 & 1 & \lambda \cos\alpha_c & 0 & 0 \end{bmatrix}. \tag{3.16}$$

The COM position can be written in terms of $\boldsymbol{q}_c$ as

$$\boldsymbol{p}_{MC} = \begin{bmatrix} x_c + Ls_c \\ y_c + Lc_c \end{bmatrix} \tag{3.17}$$

where $s_c = \sin(\alpha_c + \theta_{c2})$ and $c_c = \cos(\alpha_c + \theta_{c2})$. The Jacobian matrix of $\boldsymbol{p}_{MC}$ with respect to $\boldsymbol{q}_c$ is

$$\boldsymbol{J}_{MC} = \frac{\partial \boldsymbol{p}_{MC}}{\partial \boldsymbol{q}_c} = \begin{bmatrix} 1 & 0 & Lc_c & Lc_c & s_c \\ 0 & 1 & -Ls_c & -Ls_c & c_c \end{bmatrix}. \tag{3.18}$$

Similarly, the Jacobian matrices $\boldsymbol{J}_{SO}$, $\boldsymbol{J}_{OS}$, and $\boldsymbol{J}_{MS}$ for the supporting leg can be calculated by simply substituting $\boldsymbol{q}_s$ for $\boldsymbol{q}_c$ in the above equations.

From now on, I denote by $\dot{\boldsymbol{q}}_o$, $\dot{\boldsymbol{q}}_c$, and $\dot{\boldsymbol{q}}_s$ respectively the velocities of the cylinder, the swing leg, and the supporting leg, and use superscripts $-$ and $+$ to distinguish quantities before and after collision. Let $\boldsymbol{F}_0$, $\boldsymbol{F}_1$, $\boldsymbol{F}_2$, and $\boldsymbol{F}_3$ denote the impulses at three contacts and the hip joint, as depicted in Fig. 3.6. From the conservation of momentum I have

$$\boldsymbol{M}_O(\dot{\boldsymbol{q}}_o^+ - \dot{\boldsymbol{q}}_o^-) = \boldsymbol{J}_{FO}^T\boldsymbol{F}_0 - \boldsymbol{J}_{CO}^T\boldsymbol{F}_1 - \boldsymbol{J}_{SO}^T\boldsymbol{F}_2 \tag{3.19a}$$

$$\boldsymbol{M}_C(\dot{\boldsymbol{q}}_c^+ - \dot{\boldsymbol{q}}_c^-) = \boldsymbol{J}_{OC}^T\boldsymbol{F}_1 + \boldsymbol{J}_{MC}^T\boldsymbol{F}_3 \tag{3.19b}$$

$$\boldsymbol{M}_S(\dot{\boldsymbol{q}}_s^+ - \dot{\boldsymbol{q}}_s^-) = \boldsymbol{J}_{OS}^T\boldsymbol{F}_2 - \boldsymbol{J}_{MS}^T\boldsymbol{F}_3 \tag{3.19c}$$

where

$$\boldsymbol{M}_O = \begin{bmatrix} m_0 & 0 & 0 \\ 0 & m_0 & 0 \\ 0 & 0 & I_0 \end{bmatrix}$$

$$\boldsymbol{M}_C = \begin{bmatrix} m_1+m_2 & 0 & m_2Lc_c & m_2Ls_c & m_2s_c \\ 0 & m_1+m_2 & -m_2Ls_c & -m_2Ls_c & m_2c_c \\ m_2Lc_c & -m_2Ls_c & M_3+I_1 & M_3 & 0 \\ m_2Lc_c & -m_2Ls_c & M_3 & M_3 & 0 \\ m_2s_c & m_2c_c & 0 & 0 & m_2 \end{bmatrix}$$

and $\boldsymbol{M}_S$ has the same form as $\boldsymbol{M}_C$ by replacing s_c and c_c with $s_s = \sin(\alpha_s + \theta_{s2})$ and $c_s = \cos(\alpha_s + \theta_{s2})$, respectively.

The contact impulses $\boldsymbol{F}_0$, $\boldsymbol{F}_1$, $\boldsymbol{F}_2$ should also satisfy the friction constraint, which can be expressed as the following linear inequality constraints:

$$\boldsymbol{N}_0^T\boldsymbol{F}_0 \geq \boldsymbol{0}_{2\times1}, \;\; \boldsymbol{N}_1^T\boldsymbol{F}_1 \geq \boldsymbol{0}_{2\times1}, \;\; \boldsymbol{N}_2^T\boldsymbol{F}_2 \geq \boldsymbol{0}_{2\times1} \tag{3.20}$$

where

$$\boldsymbol{N}_0^T = \begin{bmatrix} 1 & \mu \\ -1 & \mu \end{bmatrix}$$

$$\boldsymbol{N}_1^T = \begin{bmatrix} \mu \sin\alpha_c + \cos\alpha_c & \mu\cos\alpha_c - \sin\alpha_c \\ \mu \sin\alpha_c + \cos\alpha_c & -\mu\cos\alpha_c + \sin\alpha_c \end{bmatrix}$$

$$\boldsymbol{N}_2^T = \begin{bmatrix} \mu \sin\alpha_s + \cos\alpha_s & \mu\cos\alpha_s - \sin\alpha_s \\ \mu \sin\alpha_s + \cos\alpha_s & -\mu\cos\alpha_s + \sin\alpha_s \end{bmatrix}$$

and μ is the friction coefficient.

To ensure pure rolling of the cylinder on the floor before and after collision, I have the following equations

$$\boldsymbol{J}_{FO}\dot{\boldsymbol{q}}_o^- = \boldsymbol{0}_{2\times 1} \tag{3.21a}$$

$$\boldsymbol{J}_{FO}\dot{\boldsymbol{q}}_o^+ = \boldsymbol{0}_{2\times 1}. \tag{3.21b}$$

I also require no slip at the contact between the swing leg and the cylinder after collision, i.e.,

$$\boldsymbol{J}_{CO}\dot{\boldsymbol{q}}_o^+ - \boldsymbol{J}_{OC}\dot{\boldsymbol{q}}_c^+ = \boldsymbol{0}_{2\times 1}. \tag{3.22}$$

The supporting leg also maintains a nonslip contact with the cylinder before and after collision, which implies

$$\boldsymbol{J}_{SO}\dot{\boldsymbol{q}}_o^- - \boldsymbol{J}_{OS}\dot{\boldsymbol{q}}_s^- = \boldsymbol{0}_{2\times 1} \tag{3.23a}$$

$$\boldsymbol{J}_{SO}\dot{\boldsymbol{q}}_o^+ - \boldsymbol{J}_{OS}\dot{\boldsymbol{q}}_s^+ = \boldsymbol{0}_{2\times 1}. \tag{3.23b}$$

Moreover, the linear velocities of the hip joint calculated from the swing and supporting legs must be the same, i.e.,

$$\boldsymbol{J}_{MS}\dot{\boldsymbol{q}}_s^- - \boldsymbol{J}_{MC}\dot{\boldsymbol{q}}_c^- = \boldsymbol{0}_{2\times 1} \tag{3.24a}$$

$$\boldsymbol{J}_{MS}\dot{\boldsymbol{q}}_s^+ - \boldsymbol{J}_{MC}\dot{\boldsymbol{q}}_c^+ = \boldsymbol{0}_{2\times 1}. \tag{3.24b}$$

3.4.3 Computing a Cyclic Walking Gait

Given the period T of a step and the average rolling velocity θ_0^d of the cylinder during a step, here I discuss how to compute a cyclic walking gait to meet these requirements. From (3.7) it follows that the motion of the system of the supporting leg and the cylinder during a step is determined by the initial state $\boldsymbol{x}_{s0}$. After the time T, the swing leg touches down on the cylinder and achieves a new initial state $\boldsymbol{x}_{c0}$ for the next step. To attain a cyclic gait, the two initial states $\boldsymbol{x}_{c0}$ and $\boldsymbol{x}_{s0}$ should be identical. Therefore, the problem of computing a cyclic walking gait is reduced to computing an initial state $\boldsymbol{x}_{s0}$ for the system such that after the time T, the swing leg collides with the cylinder and achieves the same initial state for the next step. In the following discussion, I first derive an expression of the error between $\boldsymbol{x}_{c0}$ and $\boldsymbol{x}_{s0}$ as a function of $\boldsymbol{x}_{s0}$. Then, I formulate an optimization problem to determine an initial state $\boldsymbol{x}_{s0}$ for a cyclic walking gait with consideration of other physical constraints.

3.4.3.1 Cost Function of the Optimization

The cost function of the optimization calculates the error between the initial states $\boldsymbol{x}_{s0}$ and $\boldsymbol{x}_{c0}$ of two successive steps. Since the state vector $\boldsymbol{x}$ comprises two components, namely the configuration component $\boldsymbol{\theta}$ and the velocity component $\dot{\boldsymbol{\theta}}$, the cost function consists of two terms that represent the errors in initial configuration and velocity between two successive steps, respectively.

To estimate the configuration error, I consider the difference in the COM position relative to the center of the cylinder at the beginning and the end of a step. If the relative

COM position at the end of a step is the same as that at the beginning of the step, then the swing foot can reach the same position as the initial position of the supporting foot of the step and the whole system can reach the same initial configuration for the next step. Hence, the configuration error can be measured by the difference in the relative COM position.

Given $\boldsymbol{x}_{s0}$, from (3.7) I can obtain the final state of the supporting leg after the time T, which is written as $\boldsymbol{x}_{sf} = \begin{bmatrix}\boldsymbol{\theta}_{sf}^T & \dot{\boldsymbol{\theta}}_{sf}^T\end{bmatrix}^T$. The position of the COM relative to the center of the cylinder at a state $\boldsymbol{x}_s$ can be calculated by

$$\boldsymbol{p}_s = \begin{bmatrix} r_0(\sin\theta_{s01} - \theta_{s1}\cos\theta_{s01}) + L\sin\theta_{s02} \\ r_0(\cos\theta_{s01} + \theta_{s1}\sin\theta_{s01}) + L\cos\theta_{s02} \end{bmatrix} \tag{3.25}$$

where $\theta_{s01} = \theta_{s0} + \theta_{s1}$ and $\theta_{s02} = \theta_{s01} + \theta_{s2}$. Let $\boldsymbol{p}_{s0}$ and $\boldsymbol{p}_{sf}$ denote the COM positions given by (3.25) at the initial state $\boldsymbol{x}_{s0}$ and the final state $\boldsymbol{x}_{sf}$, respectively. I expect $\boldsymbol{p}_{sf} = \boldsymbol{p}_{s0}$ such that $\boldsymbol{\theta}_{c0} = [\theta_{c0} \quad \theta_{c1} \quad \theta_{c2} \quad l_c]^T$ of the swing leg before and after collision can equal the initial value $\boldsymbol{\theta}_{s0}$ of the supporting leg. The error between $\boldsymbol{p}_{s0}$ and $\boldsymbol{p}_{sf}$, which gives the error between $\boldsymbol{\theta}_{s0}$ and $\boldsymbol{\theta}_{c0}$, is represented as

$$e_{\mathrm{COM}} = \frac{1}{2}(\boldsymbol{p}_{sf} - \boldsymbol{p}_{s0})^T \boldsymbol{W}_P (\boldsymbol{p}_{sf} - \boldsymbol{p}_{s0}). \tag{3.26}$$

Assume that $\boldsymbol{\theta}_{c0}$ reaches the same value as $\boldsymbol{\theta}_{s0}$. Then I expect that $\dot{\boldsymbol{\theta}}_{c0}$ is also equal to $\dot{\boldsymbol{\theta}}_{s0}$. To determine $\dot{\boldsymbol{\theta}}_{c0}$, I need to compute the collision model presented in Section 3.4.2. In computing the matrices $\boldsymbol{M}_C$, $\boldsymbol{N}_1$, $\boldsymbol{J}_{CO}$, $\boldsymbol{J}_{OC}$, $\boldsymbol{J}_{MC}$ in the collision model, I take $x_o = 0$, $y_o = 0$, $x_c = r_0\sin(\theta_{s0}+\theta_{s1}) - r_0\theta_{s1}\cos(\theta_{s0}+\theta_{s1})$, $y_c = r_0\cos(\theta_{s0}+\theta_{s1}) + r_0\theta_{s1}\sin(\theta_{s0}+\theta_{s1})$, $\alpha_c = \theta_{s0} + \theta_{s1}$, and $\theta_{c2} = \theta_{s2}$, where θ_{s0}, θ_{s1}, and θ_{s2} are the components of $\boldsymbol{\theta}_{s0}$. The computation of $\boldsymbol{M}_S$, $\boldsymbol{N}_2$, $\boldsymbol{J}_{SO}$, $\boldsymbol{J}_{OS}$, and $\boldsymbol{J}_{MS}$ is similar except that θ_{s0}, θ_{s1}, and θ_{s2} are the components of $\boldsymbol{\theta}_{sf}$.

In the collision model, the velocities $\dot{\boldsymbol{q}}_o^-$, $\dot{\boldsymbol{q}}_o^+$, $\dot{\boldsymbol{q}}_c^-$, $\dot{\boldsymbol{q}}_c^+$, $\dot{\boldsymbol{q}}_s^-$, $\dot{\boldsymbol{q}}_s^+$ and the impulses $\boldsymbol{F}_0$, $\boldsymbol{F}_1$, $\boldsymbol{F}_2$, $\boldsymbol{F}_3$ are unknown quantities that need to be determined. Here, $\dot{\boldsymbol{q}}_o^-$ and $\dot{\boldsymbol{q}}_s^-$ can be

calculated from $\boldsymbol{x}_{sf}$ as

$$\dot{\boldsymbol{q}}_o^- = \begin{bmatrix} r_0\dot{\theta}_{s0} & 0 & \dot{\theta}_{s0} \end{bmatrix}^T \tag{3.27}$$

$$\dot{\boldsymbol{q}}_s^- = \begin{bmatrix} R_{11}\dot{\theta}_{s0} + R_{12}\dot{\theta}_{s1} & R_{21}\dot{\theta}_{s0} + R_{22}\dot{\theta}_{s1} & \dot{\theta}_{s01} & \dot{\theta}_{s2} & \dot{l}_s \end{bmatrix}^T \tag{3.28}$$

where $R_{11} = r_0(1 + \cos\theta_{s01} + \theta_{s1}\sin\theta_{s01})$, $R_{12} = r_0\theta_{s1}\sin\theta_{s01}$, $R_{21} = r_0(\theta_{s1}\cos\theta_{s01} - \sin\theta_{s01})$, $R_{22} = r_0\theta_{s1}\cos\theta_{s01}$, $\theta_{s01} = \theta_{s0} + \theta_{s1}$, $\dot{\theta}_{s01} = \dot{\theta}_{s0} + \dot{\theta}_{s1}$ and θ_{s0}, θ_{s1}, θ_{s2}, $\dot{\theta}_{s0}$, $\dot{\theta}_{s1}$, and $\dot{\theta}_{s2}$ are the components of $\boldsymbol{x}_{sf}$. It can be verified that $\dot{\boldsymbol{q}}_o^-$ and $\dot{\boldsymbol{q}}_s^-$ satisfy (3.21a) and (3.23a), respectively. The other contact constraints in (3.21)–(3.24) together with (3.19) can be rewritten in the matrix form

$$\boldsymbol{Q}\dot{\boldsymbol{q}} = \boldsymbol{b} \tag{3.29}$$

where $\dot{\boldsymbol{q}} = [\dot{\boldsymbol{q}}_o^+ \quad \dot{\boldsymbol{q}}_s^+ \quad \dot{\boldsymbol{q}}_c^+ \quad \dot{\boldsymbol{q}}_c^- \quad \boldsymbol{F}_0 \quad \boldsymbol{F}_1 \quad \boldsymbol{F}_2 \quad \boldsymbol{F}_3]^T \in \mathbb{R}^{26}$, $\boldsymbol{Q} \in \mathbb{R}^{23\times 26}$, $\boldsymbol{b} \in \mathbb{R}^{23}$, and

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{M}_O & \boldsymbol{0}_{3\times5} & \boldsymbol{0}_{3\times5} & \boldsymbol{0}_{3\times5} & -\boldsymbol{J}_{FO}^T & \boldsymbol{J}_{CO}^T & \boldsymbol{J}_{SO}^T & \boldsymbol{0}_{3\times2} \\ \boldsymbol{0}_{5\times3} & \boldsymbol{0}_{5\times5} & -\boldsymbol{M}_C & \boldsymbol{M}_C & \boldsymbol{0}_{5\times2} & \boldsymbol{J}_{OC}^T & \boldsymbol{0}_{5\times2} & \boldsymbol{J}_{MC}^T \\ \boldsymbol{0}_{5\times3} & \boldsymbol{M}_S & \boldsymbol{0}_{5\times5} & \boldsymbol{0}_{5\times5} & \boldsymbol{0}_{5\times2} & \boldsymbol{0}_{5\times2} & -\boldsymbol{J}_{OS}^T & \boldsymbol{J}_{MS}^T \\ \boldsymbol{J}_{FO} & \boldsymbol{0}_{2\times5} & \boldsymbol{0}_{2\times5} & \boldsymbol{0}_{2\times5} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} \\ \boldsymbol{J}_{CO} & \boldsymbol{0}_{2\times5} & -\boldsymbol{J}_{CO} & \boldsymbol{0}_{2\times5} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} \\ \boldsymbol{J}_{SO} & -\boldsymbol{J}_{SO} & \boldsymbol{0}_{2\times5} & \boldsymbol{0}_{2\times5} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} \\ \boldsymbol{0}_{2\times3} & \boldsymbol{0}_{2\times5} & \boldsymbol{0}_{2\times5} & \boldsymbol{J}_{MC} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} \\ \boldsymbol{0}_{2\times3} & \boldsymbol{J}_{MS} & -\boldsymbol{J}_{MC} & \boldsymbol{0}_{2\times5} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} \end{bmatrix}$$

$$\boldsymbol{b} = \begin{bmatrix} \left(\boldsymbol{M}_O\dot{\boldsymbol{q}}_o^-\right)^T & \boldsymbol{0}_{1\times5} & \left(\boldsymbol{M}_s\dot{\boldsymbol{q}}_s^-\right)^T & \boldsymbol{0}_{1\times2} & \boldsymbol{0}_{1\times2} & \boldsymbol{0}_{1\times2} & \boldsymbol{0}_{1\times2} & \left(\boldsymbol{J}_{MS}\dot{\boldsymbol{q}}_s^-\right)^T \end{bmatrix}^T.$$

Equation (3.29) is underdetermined. The impulses $\boldsymbol{F}_0$, $\boldsymbol{F}_1$, $\boldsymbol{F}_2$ must also satisfy (3.20). From $\dot{\boldsymbol{q}}_0^+$ and $\dot{\boldsymbol{q}}_s^+$ I derive

$$\dot{\boldsymbol{\theta}}_{c0} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_0^+ \\ \dot{\boldsymbol{q}}_s^+ \end{bmatrix} = \boldsymbol{P}\dot{\boldsymbol{q}}. \tag{3.30}$$

I shall minimize the difference between $\dot{\boldsymbol{\theta}}_{c0}$ and $\dot{\boldsymbol{\theta}}_{s0}$, i.e.,

$$\begin{cases} \text{minimize } \frac{1}{2}\|\boldsymbol{P}\dot{\boldsymbol{q}} - \dot{\boldsymbol{\theta}}_{s0}\|^2 \\ \text{subject to } (3.20) \text{ and } (3.29). \end{cases} \tag{3.31}$$

Let e_{state} be the minimum objective value of (3.31). It gives the error between $\dot{\boldsymbol{\theta}}_{c0}$ and $\dot{\boldsymbol{\theta}}_{s0}$ after collision. Also, it gives the error between $\boldsymbol{x}_{c0}$ and $\boldsymbol{x}_{s0}$, since $\boldsymbol{\theta}_{c0}$ is taken to be $\boldsymbol{\theta}_{s0}$.

Finally, the cost function of the optimization to be minimized is

$$E = e_{\text{COM}} + e_{\text{state}}. \tag{3.32}$$

From the above arguments, E is a function of $\boldsymbol{x}_{s0}$.

3.4.3.2 Constraints of the Optimization

Now I consider a few constraints on $\boldsymbol{x}_{s0}$. First, the contact point between the supporting foot and the cylinder should stay within the sole during the entire step. Let l_h and l_t denote the distance from the ankle joint to the heel and the toe, respectively. Then θ_1 should be limited within $[-l_h/r_0, l_t/r_0]$. From (3.7) it follows $\theta_1 = \boldsymbol{a}_2^T\boldsymbol{x}_{s0}$, where $\boldsymbol{a}_2^T$ is the second row of $e^{(\boldsymbol{A}-\boldsymbol{BK})t}$. Thus, I have

$$-l_h/r_0 \le \max_{t\in[0,T]} \left|\boldsymbol{a}_2^T\boldsymbol{x}_{s0}\right| \le l_t/r_0. \tag{3.33}$$

Besides, I require the cylinder to roll at a desired average rolling velocity $\dot{\theta}_0^d$ during a step. From (3.7) this requirement can be expressed as

$$\left(\boldsymbol{a}_1^T - \boldsymbol{e}_1^T\right)\boldsymbol{x}_{s0} = \dot{\theta}_0^d T \tag{3.34}$$

where $\boldsymbol{a}_1^T$ is the first row of $e^{(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K})T}$ and $\boldsymbol{e}_1^T = [1 \quad \boldsymbol{0}_{1\times 7}]$. Equation (3.34) implies that the initial state $\boldsymbol{x}_{s0}$ for achieving a desired average velocity lies on a hyperplane with normal $\boldsymbol{a}_1 - \boldsymbol{e}_1$ in the state space.

Combining the cost function (3.32) and the constraints (3.33) and (3.34), I formulate the computation of the initial state $\boldsymbol{x}_{s0}$ for cyclic walking as the following optimization problem

$$\begin{cases} \text{minimize} \;\; E \\ \text{subject to} \;\; (3.33) \text{ and } (3.34). \end{cases} \tag{3.35}$$

I pursue $\boldsymbol{x}_{s0}$, for which the minimum value of E is zero.

3.4.4 Maintaining a Cyclic Walking Gait

Because of modeling errors and external disturbances, the state of the supporting leg at the end of a walking cycle may be different from a planned gait. As a consequence, the swing leg may not reach the desired states for a new walking cycle before and after collision. In this section, I discuss how to recompute its state so that the robot recovers a planned cyclic gait.

3.4.4.1 Inverse Kinematics

The final COM position $\boldsymbol{p}_{sf}$, which can be obtained by (3.25) with respect to the final state $\boldsymbol{x}_{sf}$ of a cycle, may slightly deviate from the initial value $\boldsymbol{p}_{s0}$. Then $\boldsymbol{\theta}_{c0}$ cannot be the same as $\boldsymbol{\theta}_{s0}$ when the swing leg touches the cylinder, and I compute the inverse kinematics with respect to $\boldsymbol{p}_{sf}$ to determine the actual $\boldsymbol{\theta}_{c0}$. Here I use the pseudoinverse method for the

inverse kinematics (Whitney, 1969), which is explained as follows. For higher numerical stability near singularities, one can use damped least squares methods (Nakamura and Hanafusa, 1986).

Similarly to (3.25), the COM position $\boldsymbol{p}_c$ with respect to $\boldsymbol{\theta}_c$ can be written as

$$\boldsymbol{p}_c = \begin{bmatrix} r_0(\sin\theta_{c01} - \theta_{c1}\cos\theta_{c01}) + L\sin\theta_{c02} \\ r_0(\cos\theta_{c01} + \theta_{c1}\sin\theta_{c01}) + L\cos\theta_{c02} \end{bmatrix} \tag{3.36}$$

where $\theta_{c01} = \theta_{c0} + \theta_{c1}$ and $\theta_{c02} = \theta_{c01} + \theta_{c2}$. Starting with an initial value of $\boldsymbol{\theta}_{c0}$, which can be taken to be $\boldsymbol{\theta}_{s0}$, the pseudoinverse method performs the following iteration to compute $\boldsymbol{\theta}_{c0}$ such that $\boldsymbol{p}_{c0} = \boldsymbol{p}_{sf}$:

$$\boldsymbol{\theta}_{c0} = \boldsymbol{\theta}_{c0} + \boldsymbol{J}^{\dagger}(\boldsymbol{p}_{sf} - \boldsymbol{p}_{c0}) \tag{3.37}$$

where $\boldsymbol{J} = \partial\boldsymbol{p}_c/\partial\boldsymbol{\theta}_c \in \mathbb{R}^{2\times 4}$ is the Jacobian matrix of $\boldsymbol{p}_c$ with respect to $\boldsymbol{\theta}_c$ and $\boldsymbol{J}^{\dagger}$ is the pseudoinverse of $\boldsymbol{J}$,

$$\boldsymbol{J} = \begin{bmatrix} r_0(1+\cos\theta_{c01}) + J_1 & J_1 & L\cos\theta_{c02} & \sin\theta_{c02} \\ J_2 - r_0\sin\theta_{c01} & J_2 & -L\sin\theta_{c02} & \cos\theta_{c02} \end{bmatrix}$$

$$J_1 = r_0\theta_{c1}\sin\theta_{c01} + L\cos\theta_{c02},\ J_2 = r_0\theta_{c1}\cos\theta_{c01} - L\sin\theta_{c02}.$$

The iteration stops once $\|\boldsymbol{J}^{\dagger}(\boldsymbol{p}_{sf} - \boldsymbol{p}_{c0})\|$ is small enough.

3.4.4.2 Initial State for the Next Step

From the inverse kinematics, I obtain the configuration component $\boldsymbol{\theta}_{c0}$, which specifies the position of the swing leg touching the cylinder. Now I determine the velocity component $\dot{\boldsymbol{\theta}}_{c0}$ after collision to obtain a complete initial state $\boldsymbol{x}_{c0}$ for the robot to walk another step.

In the determining of $\dot{\boldsymbol{\theta}}_{c0}$, first I still need to consider the collision model (3.29). Also I expect the cylinder to achieve the desired average rolling velocity $\dot{\theta}_0^d$. Thus $\boldsymbol{x}_{c0}$ should

satisfy (3.34), which can be rewritten as a linear equality constraint on $\dot{\boldsymbol{\theta}}_{c0}$:

$$\boldsymbol{a}_{12}^T \dot{\boldsymbol{\theta}}_{c0} = \dot{\theta}_0^d T - \left(\boldsymbol{a}_{11}^T - [1 \quad \mathbf{0}_{1\times 3}]\right)\boldsymbol{\theta}_{c0} \tag{3.38}$$

where $\boldsymbol{a}_{11}$ and $\boldsymbol{a}_{12}$ contain the first and last four components of $\boldsymbol{a}_1$, respectively. Combining (3.29), (3.30), and (3.38), I obtain

$$\begin{bmatrix} \boldsymbol{Q} \\ \boldsymbol{a}_{12}^T \boldsymbol{P} \end{bmatrix} \dot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{b} \\ \dot{\theta}_0^d T - \left(\boldsymbol{a}_{11}^T - [1 \quad \mathbf{0}_{1\times 3}]\right) \boldsymbol{\theta}_{c0} \end{bmatrix}. \tag{3.39}$$

Once solving (3.39) for $\dot{\boldsymbol{q}}$, I can calculate $\dot{\boldsymbol{\theta}}_{c0}$ by (3.30). Nevertheless, it should be noted that (3.39) is an underdetermined system, which has an infinite number of solutions. In order to maintain a planned cyclic gait, I pursue the solution to (3.39) that minimizes the cost function defined as follow.

First, I wish to minimize the error in the COM position at the end of each step. From (3.7) and (3.30), the final state of a step after collision can be written as

$$\boldsymbol{x}_{cf} = e^{(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K})T} \begin{bmatrix} \boldsymbol{\theta}_{c0} \\ \boldsymbol{P}\dot{\boldsymbol{q}} \end{bmatrix}. \tag{3.40}$$

Then I can compute the COM position $\boldsymbol{p}_{cf}$ at the end of the step after collision as (3.25). Since $\boldsymbol{\theta}_{c0}$ has been determined by the inverse kinematics computation, from (3.40) it follows that $\boldsymbol{x}_{cf}$ is a function of only $\dot{\boldsymbol{q}}$ and so is $\boldsymbol{p}_{cf}$. Let $\boldsymbol{x}_0^* = [\boldsymbol{\theta}^* \quad \dot{\boldsymbol{\theta}}^*]^T$ be an optimal initial state obtained by solving the optimization problem (3.35) and $\boldsymbol{p}^*$ the COM position calculated by (3.25) with respect to $\boldsymbol{x}_0^*$. Thus the error in the COM position is represented as

$$e_{\text{COM}} = \frac{1}{2}(\boldsymbol{p}_{cf} - \boldsymbol{p})^T \boldsymbol{W}_p (\boldsymbol{p}_{cf} - \boldsymbol{p}) \tag{3.41}$$

where $\boldsymbol{p} = (1 - k_p)\boldsymbol{p}_{sf} + k_p \boldsymbol{p}^*$ and $k_p \in [0, 1]$.

I also intend to minimize the error in the initial state of each step, which is represented as

$$e_{\text{state}} = \frac{1}{2}(\dot{\boldsymbol{\theta}}_{c0} - \dot{\boldsymbol{\theta}})^T \boldsymbol{W}_s (\dot{\boldsymbol{\theta}}_{c0} - \dot{\boldsymbol{\theta}}) \tag{3.42}$$

where $\dot{\boldsymbol{\theta}} = (1 - k_s)\dot{\boldsymbol{\theta}}_{s0} + k_s \dot{\boldsymbol{\theta}}^*$ and $k_s \in [0, 1]$.

Therefore, the cost function is defined as $e_{\text{COM}} + e_{\text{state}}$ and the solution for $\dot{\boldsymbol{q}}$ is reduced to the optimization problem

$$\begin{cases} \text{minimize } \ e_{\text{COM}} + e_{\text{state}} \\ \text{subject to } \ (3.39) \text{ and } (3.20). \end{cases} \tag{3.43}$$

Using larger values for k_p and k_s, the resulting gait may be closer to the cyclic gait obtained from the optimization (3.35).

3.4.5 Simulation Results

3.4.5.1 Setup for Optimization

The parameters of the simplified dynamics model are $m_0 = 157$ kg, $I_0 = 20$ kg $\cdot$ m^2, $m_1 = 4$ kg, $I_1 = 0.05$ kg $\cdot$ m^2, $m_2 = 61$ kg, $I_2 = 12$ kg $\cdot$ m^2, $r_0 = 0.5$ m, and $L_0 = 0.8$ m. I set the step time $T = 0.5$ s and the desired rolling velocity $\dot{\theta}_0^d = 0.2$ rad/s. The upper and lower bounds on $\boldsymbol{x}_0$ are $\boldsymbol{x}_0^{\text{lb}} = [-\pi/4 \ \ -0.15 \ \ -\pi/2 \ \ -\pi/2 \ \ -\pi/2 \ \ -\pi/2]^T$ and $\boldsymbol{x}_0^{\text{ub}} = [0 \ \ 0.15 \ \ \pi/2 \ \ \pi/2 \ \ \pi/2 \ \ \pi/2]^T$. The function **fmincon** provided by the Matlab Optimization Toolbox is used to solve (3.35) and (3.43).

3.4.5.2 Optimal Cyclic Gait

Fig. 3.7 shows the optimal initial states with slightly different minimized cost function values obtained by solving (3.35) with random initial values for the function **fmincon** between $\boldsymbol{x}_0^{\text{lb}}$ and $\boldsymbol{x}_0^{\text{ub}}$. Most of the optimal initial states have cost function values below 10^{-8}. In Fig. 3.8, the optimal initial states are colored according to the energy consumed in the

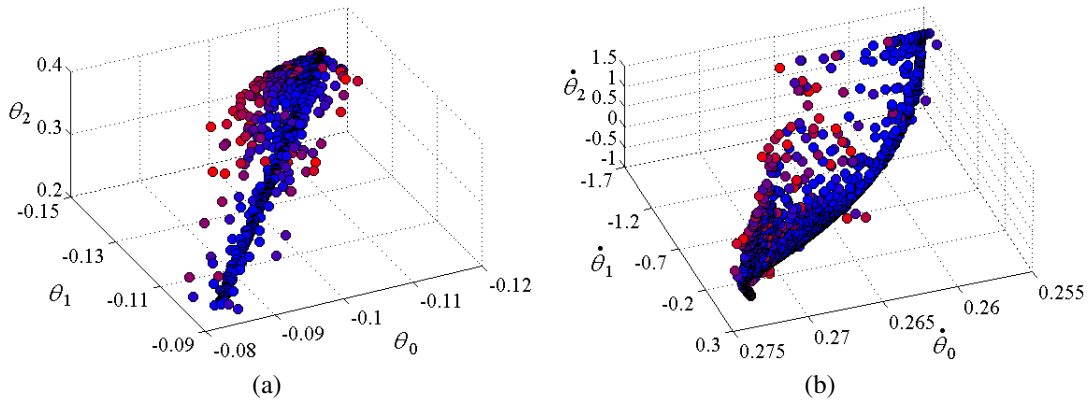


Figure 3.7. Distribution of optimal initial states in the state space. Blue and red dots represent the optimal initial sates with smaller and larger values of the cost function defined by (3.32).

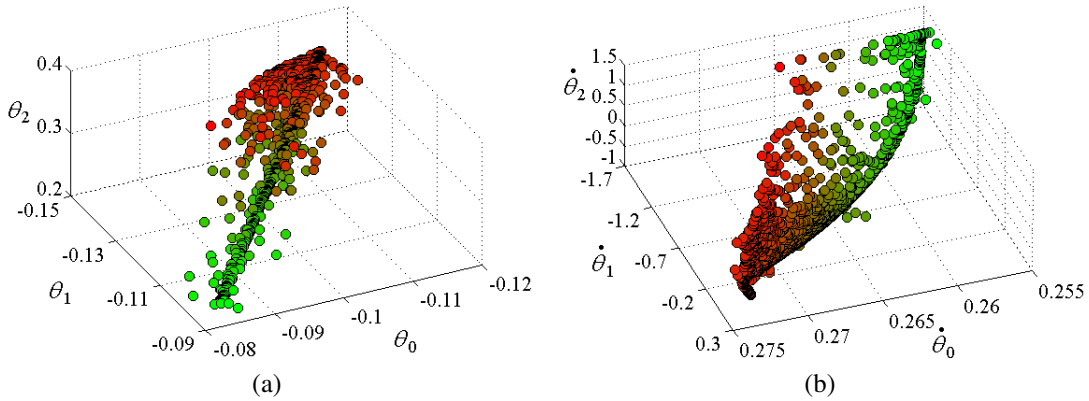


Figure 3.8. Distribution of optimal initial states with smaller (marked in green) and larger (marked in red) energy consumption for one step.

step, which is estimated by the squared sum of ankle torques at every time step. It is clear that the walking gait with smaller initial $|\theta_0 + \theta_1|$ has lower energy consumption, probably because the robot stands closer to the top of the cylinder and possesses larger potential energy. Equation (3.34) shows that the initial state lies on a hyperplane with normal $\boldsymbol{a}_1 - \boldsymbol{e}_1$. I slightly change each optimal initial state along the normal by the same amount and then plot optimal initial states in Fig. 3.9 colored according to the change of the cost function, which implies the robustness of a planned cyclic gait. It can be seen that the changes for most optimal initial states are similar.

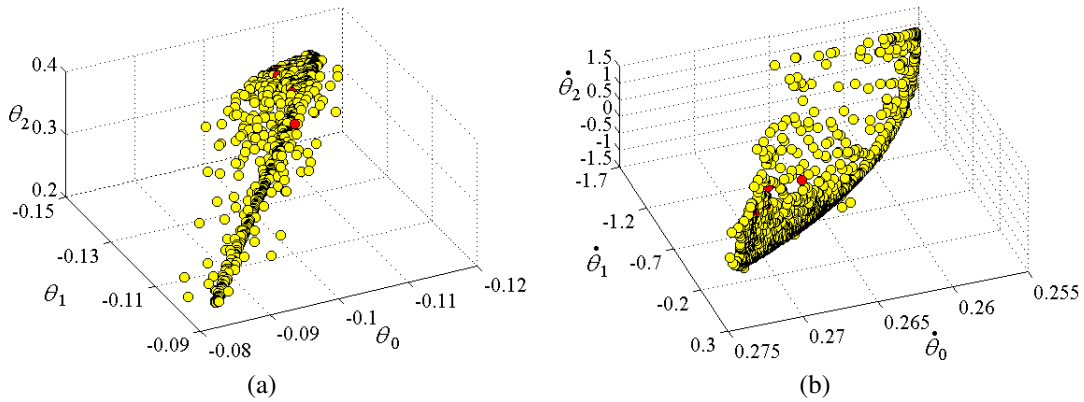


Figure 3.9. Optimal initial states colored according to the change in the cost function while shifting optimal initial states in the normal direction. The yellow (red) color means that the change is relatively smaller (bigger).

Fig. 3.10 depicts 100 step cycles starting with an arbitrary optimal initial state. Due to the numerical error in solving the optimization problem (3.35), the spring-damper motion slightly deviates from the planned motion, as shown in Fig. 3.10(d). From Fig. 3.10(a) it can be seen that the cylinder rolls 0.1 rad in one cycle, which implies that the average velocity is 0.2 rad/s and reaches the desired value, as the cycle period is 0.5 s.

3.4.5.3 Simulation Under Disturbance

I change the mass and inertia of the simulated model to $m_2 = 70$ kg and $I_2 = 15$ kg $\cdot$ m^2 to emulate the modeling error. I also add a Gaussian random error with zero mean and deviation of 0.2 N $\cdot$ m as the noise to the ankle torque at every time step. By the method proposed in Section 3.4.4 with $k_p = 0.2$ and $k_s = 0.2$, the robot can still achieve stable cycles, as shown in Fig. 3.11. The cycles are slightly different from each other and those shown in Fig. 3.10 because of the disturbances in the model and the ankle torque. Nevertheless, the average velocity remains close to the desired value.

By the proposed methods, I can achieve cyclic walking gaits with different average velocities even under larger disturbances, as shown in Fig. 3.12, where $\dot{\theta}_0^d = 0.4$ s^{-1},

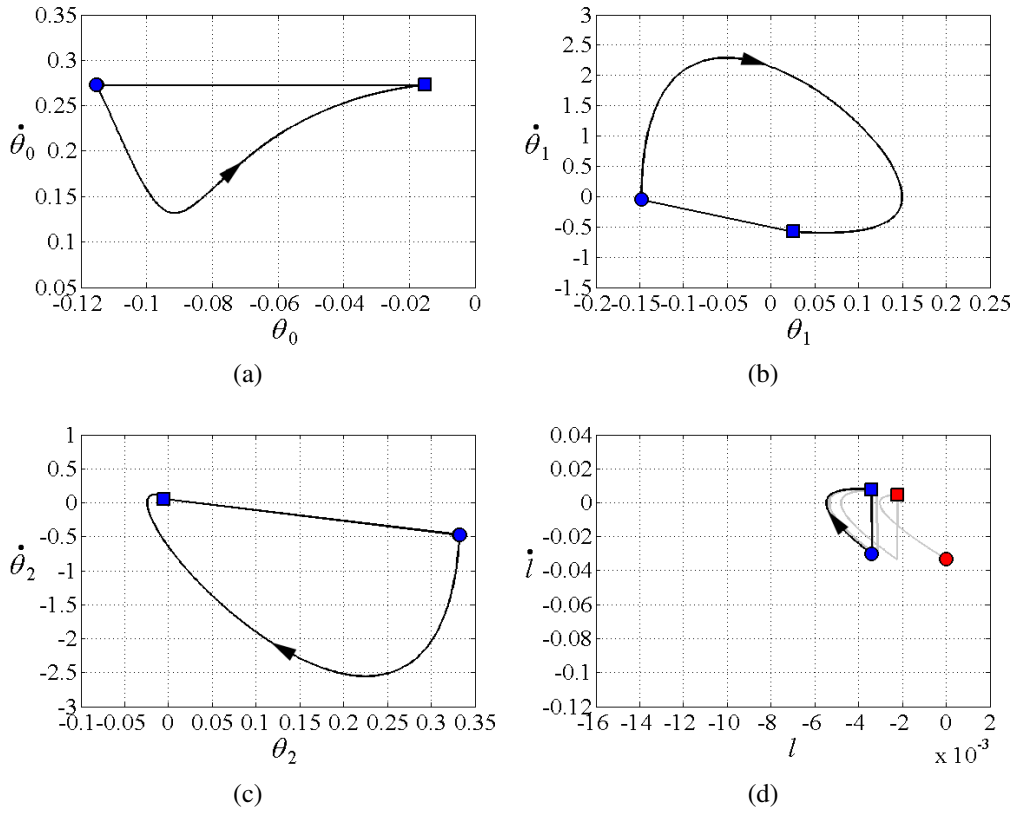


Figure 3.10. One hundred walking cycles starting with an optimal initial state. Each curve from a dot to a square represents a step. The red color denotes the first step starting with the optimal initial state, while the blue color denotes the last step.

$m_2 = 80$ kg and $I_2 = 20$ kg $\cdot$ m^2, and the deviation of the Gaussian random noise to the ankle torque is 0.5 N $\cdot$ m.

Fig. 3.13 displays the snapshots of one step of the two cyclic gaits, while a video posted on `http://www.cs.unc.edu/~yuzheng/dissertation/` exhibits 20 steps.

3.5 Conclusions and Future Work

In this chapter, I investigate the problem of generating and controlling bipedal walk on a rolling cylinder. Based on the balance control presented in the previous chapter, I first propose an approach to generating static walking gaits. Second, I derive a collision model for the supporting leg exchange and establish an optimization problem to compute the optimal initial state such that the robot can achieve a cyclic walking gait on the cylinder

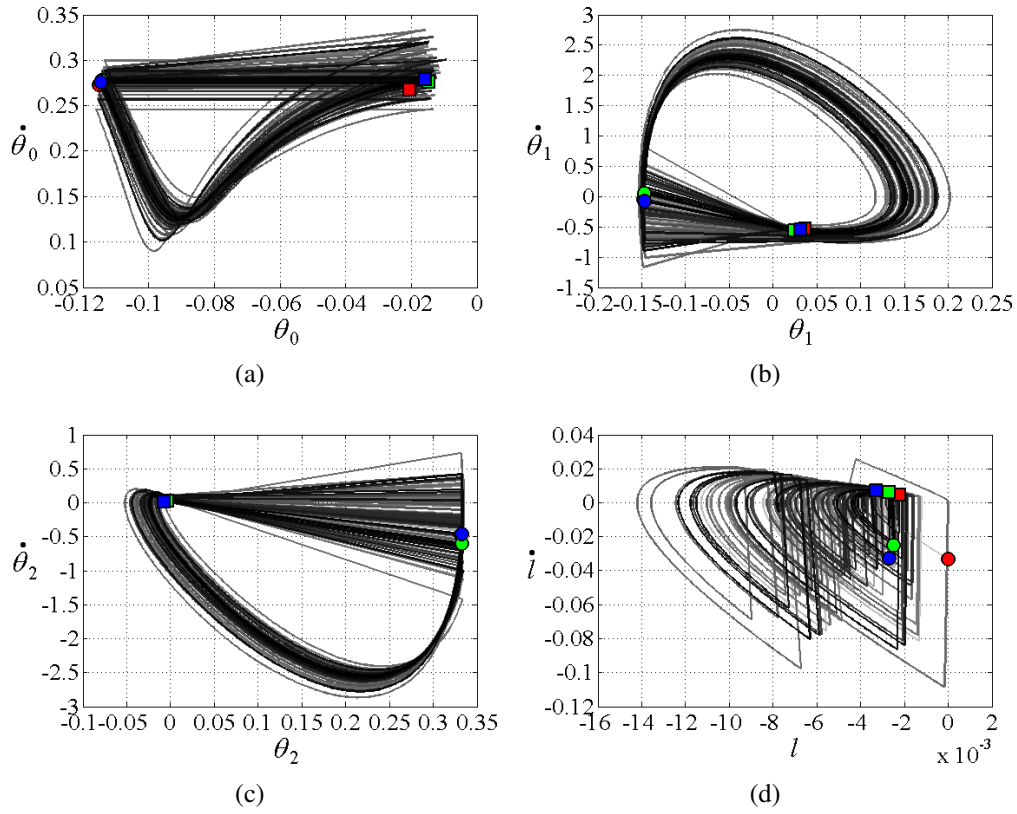


Figure 3.11. Walking cycles under disturbances in the model and the ankle torque. The red color denotes the first step starting with the optimal initial state, while the green and blue colors denote the last two steps, which are slightly different due to the random disturbance.

with a desired average rolling velocity. In consideration of modeling errors and external disturbance, I also propose a method for determining an appropriate state of the swing leg before collision to maintain the robot in a stable cyclic walking gait.

The ultimate goal is to realize the bipedal walk on a real robot. To do this, there are many other issues that need to be considered. First, I shall explore how to bring the robot to a planned optimal initial state for a cyclic walk. Second, the motion of the swing leg may cause the robot and the cylinder to deviate from the planned motion and its dynamics needs to be considered in the full-body control of the robot. Third, there are other errors, such as the modelling errors and tracking errors, such that the robot cannot perfectly follow the planned gait. All these issues need to be considered in hardware experiments on a real robot in the future.

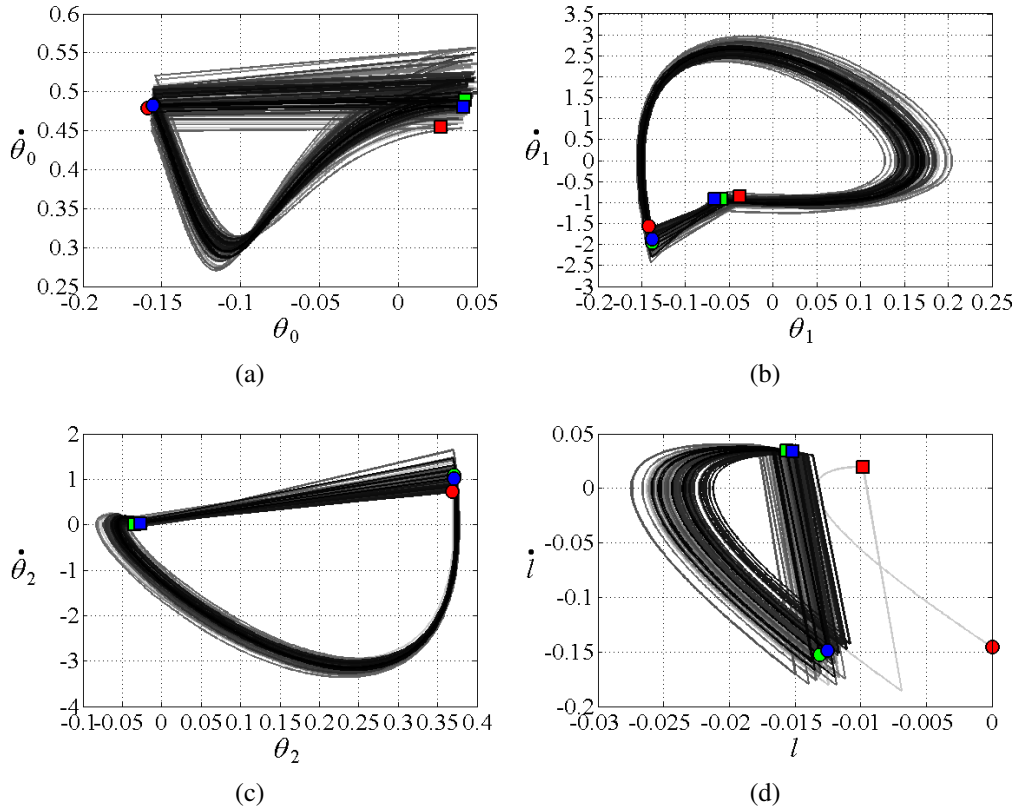
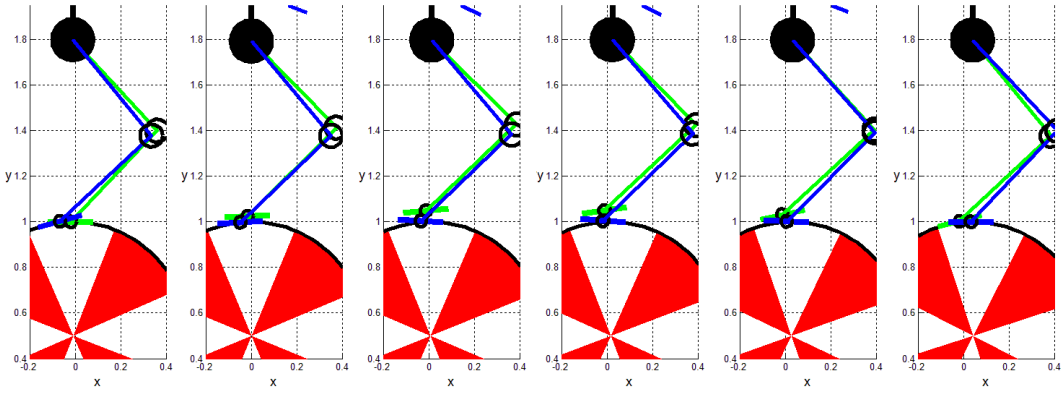
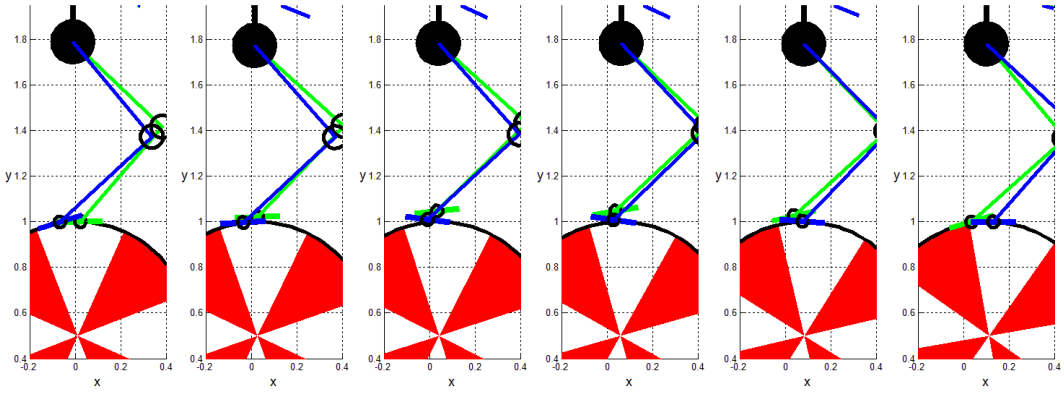


Figure 3.12. Walking cycles with desired average velocity equal to 0.4 rad/s.



(a)



(b)

Figure 3.13. Snapshots of one cyclic step with average velocity equal to (a) $0.2\ \mathrm{rad/s}$ and (b) $0.4\ \mathrm{rad/s}$ under different disturbances.

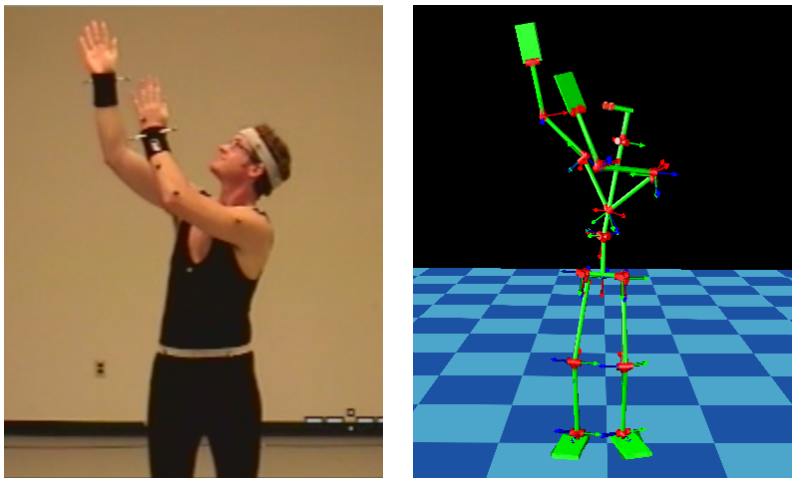
CHAPTER 4: MOTION TRACKING CONSIDERING STRICT CONTACT CONSTRAINTS

4.1 Introduction

One day in the future humanoid robots are expected to work with humans in home and office environments. It would be more desirable for such robots to have human-like motions so that human co-workers can easily infer their intention and predict future movements for safe and smooth interactions.

However, programming humanoid robots is not straightforward because they tend to have complex structures consisting of many joints. A possible solution is to teach the motions through human demonstration as often referred to as learning from demonstration (Billard et al., 2008) or imitation learning (Schaal et al., 2003). This approach allows a programmer to simply demonstrate the motion while the robot observes the motion. A learning algorithm then makes adjustments to the motion so that the robot can achieve the task using its own body.

Unfortunately, most of the work based on this approach considers only the kinematics of motions and therefore cannot be directly applied to robots and motions that require balancing, such as standing and walking motions of floating-base humanoid robots. Considering the dynamics in such motions is essential because the six DOFs of the translation and rotation of the floating base are not directly actuated. Instead, the corresponding generalized force is provided by contact forces that are subject to inequality constraints on the friction.



(a) (b)

Figure 4.1. Example of human motion tracking. (a) The original human motion. (b) The simulated robot motion.

4.1.1 Main Results

In this chapter, I discuss a controller that enables floating-base humanoid robots to track motion capture data while maintain balance, as illustrated in Fig. 4.1. Unlike the previous work with similar goals, the controller does not include a balance controller based on simplified models. Also, it takes into account the strict friction constraint on contact forces.

The controller consists of two components. The first component is a standard proportional-derivative (PD) tracking controller that computes the desired acceleration to track the given reference trajectory of every DOF, including the six unactuated ones of the floating base. The second component computes the optimal contact forces and joint torques to realize the desired accelerations given by the first component, considering the full-body dynamics of the robot and the strict friction constraints on contact forces. The desired accelerations may not be feasible for the robot due to the limits in normal contact forces and friction. Hence, it is required to compute the feasible contact forces and joint torques for the robot to realize the desired accelerations to some extent without violating the limits. I decouple the computation of contact forces and joint torques into two simple sub-problems by taking

advantage of the property that the joint torques do not contribute to the six DOFs of the floating base, which allows me to consider strict contact force constraints and accomplish the computation in real time.

Finally, I demonstrate the usefulness of the tracking controller in full-body dynamics simulation with two settings. In the first setting, I require a humanoid robot to track choreographic human motions and maintain both feet on the ground, while the feet in the motion capture data are not perfectly still due to errors in motion capturing and differences between the kinematics of the human subject and the robot. In the second, the robot is required to follow human stepping motions where the two feet lift up and touch down alternately. By using the proposed controller, the robot can successfully track the captured human motions.

4.1.2 Organization

This chapter is organized as follows. Section 4.2 briefly reviews the previous work. Section 4.3 introduces the full-body dynamics of a floating-base robot involving contact with the environment. Section 4.4 describes the motion tracking controller. Section 4.5 addresses how to compute feasible and optimal contact forces. Section 4.6 shows simulation results with various reference motions. Section 4.7 provides the conclusion and a discussion of future work.

4.2 Previous Work

Since humanoid robots have similar structures to humans, using human motion capture data to program humanoid robots seems to be an effective way to generate human-like motions. Ude et al. (2000) and Safonova et al. (2003) mapped human motions to fixed-base humanoid robots considering the kinematic constraints of the robot. Ikemata et al. (1999) and Yamane and Nakamura (2003) discussed the adapting of human motion data to the dynamics of floating-base humanoid robots. Miura et al. (2009) and Boutin et al. (2010)

developed methods for generating humanoid locomotion based on motion capture data, which modify the extracted joint trajectories according to a replanned ZMP trajectory that ensures the dynamic consistency. Nakaoka et al. (2003) proposed a method to convert human dancing motions to physically feasible motions for humanoid robots by manually segmenting a motion into motion primitives and designing a controller for each of them. However, these methods are aimed at offline planning. Some methods can realize online tracking of upper-body motions in the double-support phase while using the lower body for balancing (Zordan and Hodgins, 2002; Ott et al., 2008). Yamane and Hodgins (2009, 2010) presented controllers for humanoid robots to simultaneously track motion capture data and maintain balance.

Using human motion data to generate motion for humanoid characters has also been studied in computer graphics (Tak et al., 2000; Safonova et al., 2004; Sok et al., 2007; da Silva et al., 2008; Muico et al., 2009). Nevertheless, those approaches usually employ an extensive optimization process and cannot be applied to realtime control of humanoid robots.

4.3 Full-Body Dynamics of a Humanoid Robot

In order to better address the problem of motion tracking, I first introduce the full-body dynamics of a humanoid robot. Assume that the robot has N_J actuated joints. Then, the total DOF of the robot is $N_G = N_J + 6$ including the six unactuated DOF of the translation and rotation of the floating base. Let $\boldsymbol{q} \in \mathbb{R}^{N_G}$ denote the generalized coordinate that defines the robot configuration and assume that its first six components correspond to the translation and rotation of the floating base. Also let N_C denote the number of links in contact with the environment and $\boldsymbol{w}_i \in \mathbb{R}^6$ $(i = 1, 2, \ldots, N_C)$ the contact wrench (force and moment) applied to the i-th contact link by the environment.

The equation of motion of the robot can be written as

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{c} = \boldsymbol{N}^T\boldsymbol{\tau} + \boldsymbol{J}_C^T\boldsymbol{w} \tag{4.1}$$

where $\boldsymbol{M} \in \mathbb{R}^{N_G \times N_G}$ is the joint-space inertia matrix of the robot, $\boldsymbol{c} \in \mathbb{R}^{N_G}$ is the sum of Coriolis, centrifugal and gravity forces, and $\boldsymbol{w} = \begin{bmatrix}\boldsymbol{w}_1^T & \boldsymbol{w}_2^T & \cdots & \boldsymbol{w}_{N_C}^T\end{bmatrix}^T \in \mathbb{R}^{6N_C}$. Matrix $\boldsymbol{N} \in \mathbb{R}^{N_J \times N_G}$ maps the joint torques into the generalized forces. Since the floating base is unactuated, the first six columns of $\boldsymbol{N}$ are all zero and $\boldsymbol{N}$ has the form

$$\boldsymbol{N} = \begin{bmatrix}\boldsymbol{0}_{N_J \times 6} & \boldsymbol{I}_{N_J \times N_J}\end{bmatrix}. \tag{4.2}$$

Matrix $\boldsymbol{J} \in \mathbb{R}^{6N_C \times N_G}$ is the contact Jacobian matrix whose transpose maps the contact wrenches into the generalized forces and has the form

$$\boldsymbol{J}_C = \begin{bmatrix}\boldsymbol{J}_{C1}^T & \boldsymbol{J}_{C2}^T & \cdots & \boldsymbol{J}_{CN_C}^T\end{bmatrix}^T \tag{4.3}$$

where $\boldsymbol{J}_{Ci} \in \mathbb{R}^{6 \times N_G}$ is the Jacobian matrix of the i-th contact link's position and orientation with respect to the generalized coordinates. Let $\dot{\boldsymbol{r}}_i \in \mathbb{R}^6$ denote the linear and angular velocities of the i-th contact link and $\ddot{\boldsymbol{r}}_i$ the accelerations. Then, the relationship between $\dot{\boldsymbol{r}}_i$ and the generalized velocity $\dot{\boldsymbol{q}}$ can be written as

$$\dot{\boldsymbol{r}}_i = \boldsymbol{J}_{Ci}\dot{\boldsymbol{q}}. \tag{4.4}$$

Differentiating (4.4), I obtain the relationship between the joint and Cartesian accelerations as

$$\ddot{\boldsymbol{r}}_i = \boldsymbol{J}_{Ci}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{Ci}\dot{\boldsymbol{q}}. \tag{4.5}$$

For a given reference motion, I identify the set of contact candidate links that includes the links that may be in contact with the environment during the motion. The set typically

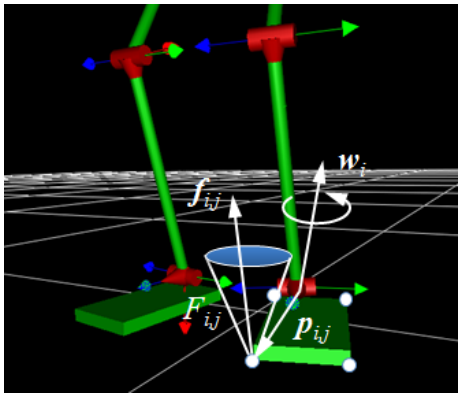


Figure 4.2. Illustration of the relationship between the contact wrench applied to a contact link and the contact forces from the environment.

consists of both feet of the robot. At any time during the motion, a contact candidate link has one of the following contact states: face contact, edge contact, point contact, or no contact.

Let $\boldsymbol{p}_{i,j} \in \mathbb{R}^3$ $(j = 1, 2, \ldots, N_i)$ be the vertices of the contact area between the i-th contact link and the environment, and $\boldsymbol{f}_{i,j} \in \mathbb{R}^3$ the contact force at the j-th contact vertex, as depicted in Fig. 4.2. The number of contact vertices, N_j, is 0, 1, 2, and equal to or greater than 3 in no, point, edge, and face contact states respectively. I assume that the contacts are rigid subject to Coulomb friction model. Therefore, $\boldsymbol{f}_{i,j}$ is a pure force and can be decomposed into three components $f_{i,j1}, f_{i,j2}, f_{i,j3}$ along the normal and two orthogonal tangential vectors at the contact vertex. Here, I do not consider slipping contact, so each contact force $\boldsymbol{f}_{i,j}$ must satisfy the friction constraint

$$F_{i,j} = \left\{ \boldsymbol{f}_{i,j} \in \mathbb{R}^3 \mid f_{i,j1} \geq 0,\ \sqrt{f_{i,j2}^2 + f_{i,j3}^2} \leq \mu_i f_{i,j1} \right\}. \tag{4.6}$$

The wrench $\boldsymbol{w}_i$ applied to the i-th contact link by the environment is the resultant force and moment from all contact forces $\boldsymbol{f}_{i,j}$ $(j = 1, 2, \ldots, N_i)$ exerted on it and can be written as

$$\boldsymbol{w}_i = \sum_{j=1}^{N_i} \boldsymbol{R}_{i,j} \boldsymbol{f}_{i,j} = \boldsymbol{R}_i \boldsymbol{f}_i \tag{4.7}$$

where matrix $\boldsymbol{R}_{i,j} \in \mathbb{R}^{6\times 3}$ maps $\boldsymbol{f}_{i,j}$ into the force and moment around the local frame of the i-th contact link where $\boldsymbol{w}_i$ is expressed, $\boldsymbol{R}_i = [\boldsymbol{R}_{i,1} \;\; \boldsymbol{R}_{i,2} \;\; \cdots \;\; \boldsymbol{R}_{i,N_i}] \in \mathbb{R}^{6\times 3N_i}$, and $\boldsymbol{f}_i = \left[\boldsymbol{f}_{i,1}^T \;\; \boldsymbol{f}_{i,2}^T \;\; \cdots \;\; \boldsymbol{f}_{i,N_i}^T\right]^T \in \mathbb{R}^{3N_i}$ comprises all contact forces applied to the i-th contact link.

The controller requires the contact vertex positions and the contact states in order to compute $\boldsymbol{R}_i$. The actual contact vertex positions and states may be different from those in the reference motion. In implementation, I use the actual contact vertex positions to compute feasible contact forces at the current pose, while using the contact states in the reference motion for the reason illustrated by the following example. Consider a case where the right foot is about to touch down. If I use the actual contact state, the optimized COP will stay in the left foot and therefore the right foot may not touch down unless the position tracking is perfect. If I use the contact state in the reference motion, on the other hand, the optimized COP will leave the left foot, which forces the right foot to touch down.

4.4 Motion Tracking Controller

Figure 4.3 shows an overview of the motion tracking controller, which consists of two major components, namely a PD controller for determining desired joint and contact link accelerations based on the reference motion and the state of the floating-base robot and an optimization module for computing required contact wrenches and joint torques to generate the desired joint and contact link accelerations.

The optimization module comprises two sequential steps as shown in the dashed box in Fig. 4.3. The first step computes the contact forces that satisfy the friction constraint and respect the reference motion as much as possible. The contact points are determined using the desired contact state from the reference motion and the current contact link positions of the actual robot. Based on the computed contact forces, the second step determines the joint torques to be used to control the robot.

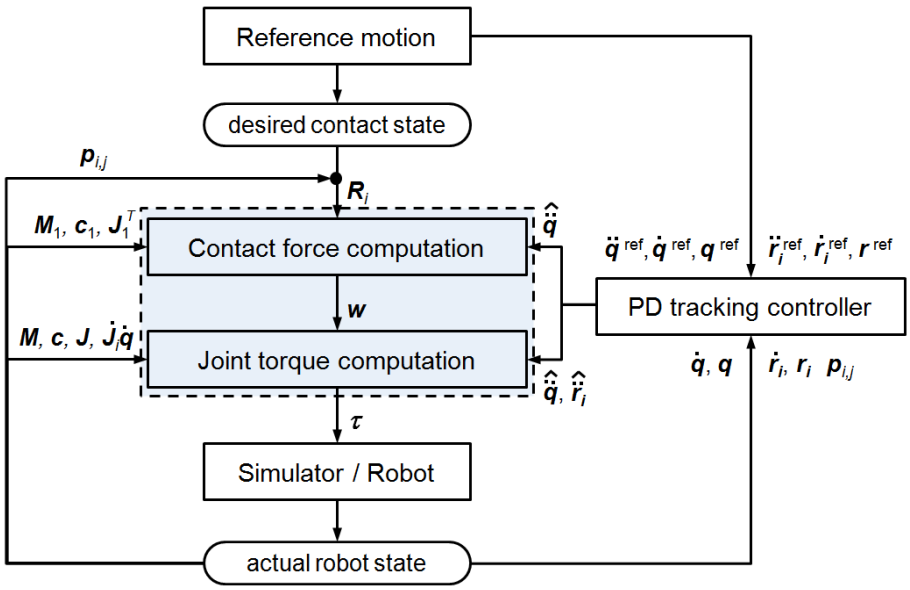


Figure 4.3. Overview of the motion tracking controller consisting of two main steps in the dashed box: 1) computation of the contact wrenches to realize the desired floating-base motion, and 2) computation of the joint torques to realize the desired full-body motion. The symbols are defined in the text.

4.4.1 Proportional-Derivative (PD) Controller

The desired accelerations of every joint are calculated based on the reference and current positions and velocities as well as the reference accelerations as

$$\ddot{\hat{\boldsymbol{q}}} = \ddot{\boldsymbol{q}}^{ref} + k_d(\dot{\boldsymbol{q}}^{ref} - \dot{\boldsymbol{q}}) + k_p(\boldsymbol{q}^{ref} - \boldsymbol{q}) \tag{4.8}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ are the current joint angle and velocity, $\boldsymbol{q}^{ref}$, $\dot{\boldsymbol{q}}^{ref}$, $\ddot{\boldsymbol{q}}^{ref}$ are the reference joint angle, velocity, and acceleration, and k_p and k_d are proportional and derivative gains.

The desired acceleration of contact candidate link i, $\ddot{\hat{\boldsymbol{r}}}_i$, is determined depending on its desired contact state. If the desired contact state is face contact, $\ddot{\hat{\boldsymbol{r}}}_i = \boldsymbol{0}$. If the desired contact state is no contact, $\ddot{\hat{\boldsymbol{r}}}_i$ is determined using the same control law as $\ddot{\hat{\boldsymbol{q}}}$:

$$\ddot{\hat{\boldsymbol{r}}}_i = \ddot{\boldsymbol{r}}_i^{ref} + k_{dc}(\dot{\boldsymbol{r}}_i^{ref} - \dot{\boldsymbol{r}}_i) + k_{pc}(\boldsymbol{r}_i^{ref} - \boldsymbol{r}_i). \tag{4.9}$$

where k_{pc} and k_{dc} are the proportional and derivative gains.

If the desired contact state is edge or point contact, I first compute the temporary desired link acceleration $\hat{\ddot{\boldsymbol{r}}}_{i0}$ by (4.9). However, $\hat{\ddot{\boldsymbol{r}}}_{i0}$ may not be consistent with the desired contact state, so the following modification is needed. I project $\hat{\ddot{\boldsymbol{r}}}_{i0}$ onto the subspace of link acceleration that satisfies the kinematic constraints of edge or point contact to obtain the desired link acceleration $\hat{\ddot{\boldsymbol{r}}}_{i0}$. If the contact link rotates around an edge, $\ddot{\boldsymbol{r}}_i$ should have the form

$$\ddot{\boldsymbol{r}}_i = \begin{bmatrix} [\boldsymbol{p}_{i,1}\times]\,(\boldsymbol{p}_{i,1} - \boldsymbol{p}_{i,2}) \\ \boldsymbol{p}_{i,1} - \boldsymbol{p}_{i,2} \end{bmatrix} \dot{\omega} \tag{4.10}$$

where $[\boldsymbol{p}_{i,1}\times] \in \mathbb{R}^{3\times 3}$ is the skew-symmetric matrix representing the cross product of $\boldsymbol{p}_{i,1}$ with another vector and $\dot{\omega} \in \mathbb{R}$ is the angular acceleration of the contact link about the edge. If the contact link rotates about a vertex, $\ddot{\boldsymbol{r}}_i$ should have the form

$$\ddot{\boldsymbol{r}}_i = \begin{bmatrix} [\boldsymbol{p}_{i,1}\times] \\ \boldsymbol{I}_{3\times 3} \end{bmatrix} \dot{\boldsymbol{\omega}} \tag{4.11}$$

where $\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ consists of the angular accelerations about the vertex. I project $\hat{\ddot{\boldsymbol{r}}}_{i0}$ onto the subspace represented by either (4.10) or (4.11) depending on whether the desired contact state is edge or point to obtain $\hat{\ddot{\boldsymbol{r}}}_i$.

4.4.2 Joint Torque Optimization Module

The role of the joint torque optimization module is to determine the joint torques such that the robot can replay a given reference motion that may or may not be physically feasible for the robot. For floating-base humanoid robots, the motion depends not only on the joint torques but also the reaction contact forces from the environment, which are also affected by the applied joint torques. Furthermore, contact forces are subject to the nonlinear friction constraints described by (4.6). Solving for the contact forces and joint torques simultaneously is computationally expensive and not suitable for realtime control.

To solve this issue, I decouple the contact forces from the joint torques by taking advantage of the property that joint torques do not affect the motion of the floating base. The optimization module therefore consists of two steps: 1) optimization of contact forces considering the friction constraints, and 2) optimization of joint torques considering the contact link acceleration constraint.

4.4.2.1 Step 1—Computing contact forces

The first six equations in the full-body dynamics equation (4.1) describe the motion of the floating base. From (4.2) it can be noticed that the six equations do not contain joint torques, which corresponds to the fact that the total linear and angular momenta are affected only by external contact forces. Extracting the first six equations of (4.1), I obtain

$$\boldsymbol{M}_1\ddot{\boldsymbol{q}} + \boldsymbol{c}_1 = \boldsymbol{J}_1^T\boldsymbol{w} \tag{4.12}$$

where $\boldsymbol{M}_1 \in \mathbb{R}^{6\times N_G}$ and $\boldsymbol{J}_1^T \in \mathbb{R}^{6\times 6N_C}$ consist of the first six rows of $\boldsymbol{M}$ and $\boldsymbol{J}^T$, respectively, and $\boldsymbol{c}_1$ comprises the first six components of $\boldsymbol{c}$. Substituting (4.7) into (4.12) yields

$$\boldsymbol{M}_1\ddot{\boldsymbol{q}} + \boldsymbol{c}_1 = \boldsymbol{G}\boldsymbol{f} \tag{4.13}$$

where $\boldsymbol{f} = \begin{bmatrix}\boldsymbol{f}_1^T & \boldsymbol{f}_2^T & \cdots & \boldsymbol{f}_{N_C}^T\end{bmatrix}^T = \begin{bmatrix}\boldsymbol{f}_{1,1}^T & \cdots & \boldsymbol{f}_{i,j}^T & \cdots & \boldsymbol{f}_{N_C,N_{N_C}}^T\end{bmatrix}^T \in \mathbb{R}^{3N_f}$ consists of the contact forces applied to all contact links and is called the total contact force, $\boldsymbol{G} = [\boldsymbol{G}_1 \;\; \boldsymbol{G}_2 \;\; \cdots \;\; \boldsymbol{G}_{N_C}] \in \mathbb{R}^{6\times 3N_f}$ with $\boldsymbol{G}_i = \boldsymbol{J}_{1i}^T\boldsymbol{R}_i \in \mathbb{R}^{6\times 3N_i}$ maps all contact forces to the wrench around the floating base and is called the total contact mapping, and $N_f = \sum_{i=1}^{N_C} N_i$ is the number of contact forces/points over all contact links. The right-hand side of (4.13) gives the resultant wrench applied to the floating base by all contact forces.

Replacing $\ddot{\boldsymbol{q}}$ with $\hat{\ddot{\boldsymbol{q}}}$ given by (4.8) on the left-hand side of (4.13), I obtain $\boldsymbol{M}_1\hat{\ddot{\boldsymbol{q}}} + \boldsymbol{c}_1$, which is the wrench at the floating base required to generate the reference motion. Because of the limitation of contact forces, however, there may not exist contact forces $\boldsymbol{f}_{i,j} \in F_{i,j}$

to generate $\boldsymbol{M}_1\hat{\ddot{\boldsymbol{q}}} + \boldsymbol{c}_1$, which implies that the desired joint accelerations $\hat{\ddot{\boldsymbol{q}}}$ may not be feasible. In order to obtain joint accelerations that are as close as possible to $\hat{\ddot{\boldsymbol{q}}}$, I formulate an optimization problem to compute the contact forces $\boldsymbol{f}_{i,j}$ described by

$$\begin{cases} \text{minimize } \|\boldsymbol{M}_1\hat{\ddot{\boldsymbol{q}}} + \boldsymbol{c}_1 - \boldsymbol{G}\boldsymbol{f}\|^2 \\ \text{subject to } \boldsymbol{f}_{i,j} \in F_{i,j} \text{ for } \forall\, i \text{ and } j \end{cases} \tag{4.14}$$

where $\|\cdot\|$ denotes the Euclidean norm. Problem (4.14) is a quadratic program with second-order cone constraints, for which efficient algorithms are available. Specific algorithms will be discussed in detail in Section 4.5. After solving (4.14) for the optimized contact forces $\boldsymbol{f}^*$, I compute the resulting contact wrenches, $\boldsymbol{w}^*$, using (4.7).

4.4.2.2 Step 2—Computing joint torques

After determining and substituting the optimized contact wrenches $\boldsymbol{w}^*$ into (4.1), I continue to compute the joint torques $\boldsymbol{\tau}$ based on the full-body dynamics. This step can be formulated as the following quadratic program

$$\begin{cases} \text{minimize } \frac{1}{2}(\ddot{\boldsymbol{q}} - \hat{\ddot{\boldsymbol{q}}})^T\boldsymbol{W}_q(\ddot{\boldsymbol{q}} - \hat{\ddot{\boldsymbol{q}}}) + \frac{1}{2}\boldsymbol{\tau}^T\boldsymbol{W}_\tau\boldsymbol{\tau} \\ \text{subject to } \boldsymbol{M}\ddot{\boldsymbol{q}} - \boldsymbol{N}^T\boldsymbol{\tau} = \boldsymbol{J}_C^T\boldsymbol{w}^* - \boldsymbol{c} \text{ and } \boldsymbol{J}_C\ddot{\boldsymbol{q}} = \hat{\dot{\boldsymbol{r}}} - \dot{\boldsymbol{J}}_C\dot{\boldsymbol{q}} \end{cases} \tag{4.15}$$

where $\ddot{\boldsymbol{q}}$ and $\boldsymbol{\tau}$ are the $N_V = N_G + N_J$ unknown variables that need to be determined and $\hat{\dot{\boldsymbol{r}}} = \begin{bmatrix} \hat{\dot{\boldsymbol{r}}}_1^T & \hat{\dot{\boldsymbol{r}}}_2^T & \cdots & \hat{\dot{\boldsymbol{r}}}_{N_C}^T \end{bmatrix}^T \in \mathbb{R}^{6N_C}$ comprises the desired accelerations for all contact links. The cost function of (4.15) consists of the error from the desired joint accelerations and the magnitude of joint torques. The first constraint is the full-body motion equation (4.1) of the robot, from which I obtain N_G linear equality constraints. After considering the full-body dynamics, therefore, N_J variables among $\ddot{\boldsymbol{q}}$ and $\boldsymbol{\tau}$ are free, which leaves me the freedom to choose or optimize the joint torques for realizing the desired full-body motion. Also, this allows me to add the second constraint, which requires that joint accelerations $\ddot{\boldsymbol{q}}$ and the

desired contact link accelerations $\hat{\ddot{\boldsymbol{r}}}_i$ to satisfy the relation (4.5), as long as $6N_C \leq N_J$. If $6N_C > N_J$, then problem (4.15) is overdetermined and may not have a feasible solution. In that case, the second constraint can be converted to a penalty term and added to the cost function. Furthermore, when a candidate contact link is in the air, the second constraint for the link may be omitted because I no longer have to constrain its motion.

I currently deal with the torque limit by adding it as a penalty term in the cost function of (4.15) rather than as inequality constraints, so that (4.15) is a quadratic optimization problem with only linear equality constraints and I can derive a closed-form solution to it. I give the derivation for the case of $6N_C \leq N_J$ as follows, while that for the case of $6N_C > N_J$ is similar.

The cost function of (4.15) can be rewritten as $\frac{1}{2}(\boldsymbol{x} - \hat{\boldsymbol{x}})^T \boldsymbol{W}(\boldsymbol{x} - \hat{\boldsymbol{x}})$, where $\boldsymbol{x} = \begin{bmatrix}\ddot{\boldsymbol{q}}^T & \boldsymbol{\tau}^T\end{bmatrix}^T \in \mathbb{R}^{N_V}$, $\hat{\boldsymbol{x}} = \begin{bmatrix}\hat{\ddot{\boldsymbol{q}}}^T & \boldsymbol{0}\end{bmatrix}^T \in \mathbb{R}^{N_V}$, and $\boldsymbol{W} = \mathrm{diag}(\boldsymbol{W}_q, \boldsymbol{W}_\tau) \in \mathbb{R}^{N_V \times N_V}$. Also, the constraints of (4.15) can be integrated as the following linear equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{4.16}$$

where

$$\boldsymbol{A} = \begin{bmatrix}\boldsymbol{M} & -\boldsymbol{N} \\ \boldsymbol{J} & \boldsymbol{0}\end{bmatrix} \in \mathbb{R}^{(N_G+6N_C)\times N_V}$$

$$\boldsymbol{b} = \begin{bmatrix}\boldsymbol{J}_C^T \boldsymbol{w}^* - \boldsymbol{c} \\ \hat{\dot{\boldsymbol{r}}} - \dot{\boldsymbol{J}}_C \dot{\boldsymbol{q}}\end{bmatrix} \in \mathbb{R}^{N_G+6N_C}.$$

Let $\boldsymbol{y} = \boldsymbol{W}^{\frac{1}{2}}(\boldsymbol{x} - \hat{\boldsymbol{x}})$. Then, the cost function of (4.15) can be further reduced to $\frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}$ and

$$\boldsymbol{x} = \boldsymbol{W}^{-\frac{1}{2}}\boldsymbol{y} + \hat{\boldsymbol{x}}. \tag{4.17}$$

Substituting (4.17) into (4.16) yields

$$\boldsymbol{A}\boldsymbol{W}^{-\frac{1}{2}}\boldsymbol{y} = \boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}. \tag{4.18}$$

Then, the optimal value of $\boldsymbol{y}$ that satisfies (4.18) and minimizes $\frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}$ can be calculated by

$$\boldsymbol{y}^* = \boldsymbol{W}^{-\frac{1}{2}}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{W}^{-1}\boldsymbol{A}^T\right)^{-1}(\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}). \tag{4.19}$$

Substituting (4.19) into (4.17), I finally obtain a closed-form solution to (4.15) as

$$\boldsymbol{x}^* = \boldsymbol{W}^{-1}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{W}^{-1}\boldsymbol{A}^T\right)^{-1}(\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}) + \hat{\boldsymbol{x}}. \tag{4.20}$$

4.5 Computation of Contact Forces

Since problem (4.15) has only linear equality constraints, I can derive a closed-form solution such that its computation becomes straightforward. By contrast, however, the computation of problem (4.14) is much more complicated because of the nonlinear inequality friction constraints, and the efficiency of the algorithm used to compute it will significantly affect the realtime applicability of the controller. Here I present efficient algorithms to solve problem (4.14) and compute feasible/minimum contact forces.

Problem (4.14) has the following geometric meaning. Let V be the set of all wrenches that can be generated on the floating base by the contact forces satisfying the friction constraint (4.6). Then, V can be written as

$$V \triangleq \left\{\boldsymbol{G}\boldsymbol{f} \in \mathbb{R}^6 \mid \boldsymbol{f}_i \in F_i \text{ for } \forall\, i\right\}. \tag{4.21}$$

For simplicity, I shorten the subscript i, j (which refers to the j-th contact force on the i-th contact link) to a single letter i and use it to represent a contact force on any contact link. Geometrically, each friction cone F_i given by (4.6) is a convex cone. Thus, it can be proved

that the set V is a convex cone in $\mathbb{R}^6$. Let

$$\boldsymbol{w} \triangleq \boldsymbol{M}_1 \dot{\hat{\boldsymbol{q}}} + \boldsymbol{c}_1 \tag{4.22}$$

which is the wrench around the floating base required to reproduce the reference motion. From (4.21) and (4.22), (4.14) can then be rewritten as

$$\begin{cases} \text{minimize} \ \ \|\boldsymbol{w} - \boldsymbol{v}\|^2 \\ \text{subject to} \ \ \boldsymbol{v} \in V \end{cases} \tag{4.23}$$

This implies that problem (4.14) is to compute the minimum Euclidean distance between $\boldsymbol{w}$ (as a point in $\mathbb{R}^6$) and the convex cone V. The point $\boldsymbol{w}$ can be contained inside or outside of V. If $\boldsymbol{w}$ is contained in V, this means that the required wrench w can be generated by the contact forces without violating the friction constraints. If not, then the required wrench $\boldsymbol{w}$ and even the desired joint accelerations given by (4.8) cannot be realized because of the limitation on the contact forces induced by the friction constraint (4.6). In that case, I compute feasible contact forces that satisfy (4.6) to generate $\boldsymbol{w}$ as much as possible. By doing this, I expect that the realized joint accelerations are close to the desired values and the resulting motion can remain similar to the reference.

4.5.1 Basic Formulations

Before getting into the detail of the algorithm for feasible/minimum contact forces, I introduce some basic definitions and formulations that will be used in the derivation hereinafter.

Prior to minimizing the contact forces, I need to define their magnitude. The magnitude of a contact force $\boldsymbol{f}_i$ can be defined as one of the following quantities:

$$\|\boldsymbol{f}_i\| \triangleq f_{i1} \tag{4.24a}$$

$$\|\boldsymbol{f}_i\| \triangleq \sqrt{f_{i1}^2 + f_{i2}^2 + f_{i3}^2}. \tag{4.24b}$$

The definition (4.24a) uses only the normal contact force, as the tangential (friction) force is bounded by the normal force according to the friction constraint (4.6). The definition (4.24b) adopts the length of the force vector, which is the conventional definition of force magnitude. The overall magnitude of a total contact force $\boldsymbol{f}$ can be defined in one of the following forms:

$$\sigma \triangleq \sum_{i=1}^{N_f} \|\boldsymbol{f}_i\| \tag{4.25a}$$

$$\sigma \triangleq \max_{i=1,2,\ldots,N_f} \|\boldsymbol{f}_i\|. \tag{4.25b}$$

A contact force having unit magnitude is called a unit contact force. A unit contact force in F_i is said to be primitive. Let U_i be the set of primitive contact forces at contact point i:

$$U_i \triangleq \big\{ \boldsymbol{f}_i \in F_i \mid \|\boldsymbol{f}_i\| = 1 \big\}. \tag{4.26}$$

It should be noted that F_i is the convex cone of U_i.

The wrench around the floating base generated by a primitive contact force is called a primitive wrench. Then, a primitive contact wrench set, which consists of all wrenches that are generated by forces in a primitive contact force set U_i, can be written as

$$W_i \triangleq \boldsymbol{G}_i(U_i). \tag{4.27}$$

Let W be the set of all resultant wrenches around the floating base that are generated by total contact forces satisfying the friction constraint with unit overall magnitude. Then, W can be formulated as

$$W \triangleq \left\{ \boldsymbol{w} = \sum_{i=1}^{N_f} \boldsymbol{G}_i \boldsymbol{f}_i \mid \sigma = 1,\ \boldsymbol{f}_i \in F_i \text{ for } \forall i \right\}. \tag{4.28}$$

Corresponding to the two definitions of σ given by (4.25), W can be rewritten as

$$W = \mathrm{CH}\left(\bigcup_{i=1}^{N_f} W_i \right) \tag{4.29a}$$

$$W = \mathrm{CH}\left(\bigoplus_{i=1}^{N_f} W_i \right) \tag{4.29b}$$

where $\mathrm{CH}(\cdot)$ denotes the convex hull of a set and $\cup$ and $\oplus$ denote the union and the Minkowski sum of sets, respectively. The convex cone defined by (4.21) can be rewritten as

$$V = \mathrm{CO}(W) \triangleq \left\{ \sum_{l=1}^{L} c_l \boldsymbol{w}_l \mid \boldsymbol{w}_l \in W \text{ and } c_l \geq 0 \text{ for } \forall\, l,\ L \in \mathbb{N}^+ \right\} \tag{4.30}$$

where $\mathrm{CO}(\cdot)$ denotes the convex cone of a set.

In the following derivation of algorithms, I will use two important functions defined on W, the support function h_W and the support mapping $\boldsymbol{s}_W$ of W, which are defined as

$$h_W(\boldsymbol{u}) \triangleq \max_{\boldsymbol{a} \in W} \boldsymbol{u}^T \boldsymbol{a}, \quad \boldsymbol{s}_W(\boldsymbol{u}) \triangleq \arg\max_{\boldsymbol{a} \in W} \boldsymbol{u}^T \boldsymbol{a} \tag{4.31}$$

where $\boldsymbol{u}$ is an arbitrary vector in $\mathbb{R}^n$. To facilitate their computation, closed-form expressions of h_W and $\boldsymbol{s}_W$ are derived as follows (Zheng and Chew, 2009; Zheng and Qian, 2009).

Corresponding to the two formulations of W in (4.29), I first derive

$$h_W(\boldsymbol{u}) = h_{W_{i^*}}(\boldsymbol{u}), \quad \boldsymbol{s}_W(\boldsymbol{u}) = \boldsymbol{s}_{W_{i^*}}(\boldsymbol{u}) \tag{4.32a}$$

$$h_W(\boldsymbol{u}) = \sum_{i \in I^*} h_{W_i}(\boldsymbol{u}), \quad \boldsymbol{s}_W(\boldsymbol{u}) = \sum_{i \in I^*} \boldsymbol{s}_{W_i}(\boldsymbol{u}) \tag{4.32b}$$

where i^* is a contact index such that $h_{W_{i^*}}(\boldsymbol{u})$ is maximal among $h_{W_i}(\boldsymbol{u})$, $i = 1, 2, \ldots, m$ and I^* is a contact index set such that $\sum_{i \in I^*} h_{W_i}(\boldsymbol{u})$ is maximal among all combinatorial sums of $h_{W_i}(\boldsymbol{u})$, $i = 1, 2, \ldots, m$. From $W_i \triangleq \boldsymbol{G}_i(U_i)$ I obtain

$$h_{W_i}(\boldsymbol{u}) = h_{U_i}(\boldsymbol{d}_i), \quad \boldsymbol{s}_{W_i}(\boldsymbol{u}) = \boldsymbol{G}_i \boldsymbol{s}_{U_i}(\boldsymbol{d}_i) \tag{4.33}$$

where $\boldsymbol{d}_i = \boldsymbol{G}_i^T \boldsymbol{u}$. The values of $h_{U_i}(\boldsymbol{d}_i)$ and $\boldsymbol{s}_{U_i}(\boldsymbol{d}_i)$ depends on which definition of $\|\boldsymbol{f}_i\|$ in (4.24) is adopted.

In the case where $\|\boldsymbol{f}_i\|$ is defined as (4.24a), I have

$$h_{U_i}(\boldsymbol{d}_i) = d_{i1} + \mu_i h_i, \quad \boldsymbol{s}_{U_i}(\boldsymbol{d}_i) = \frac{1}{h_i} \begin{bmatrix} h_i & \mu_i d_{i2} & \mu_i d_{i3} \end{bmatrix}^T \tag{4.34}$$

where $h_i = \sqrt{d_{i2}^2 + d_{i3}^2}$. If $h_i = 0$, then $\boldsymbol{s}_{U_i}(\boldsymbol{d}_i)$ can be any point in U_i.

In the case where $\|\boldsymbol{f}_i\|$ is defined as (4.24b), if $h_i \leq \mu_i d_{i1}$, I derive

$$h_{U_i}(\boldsymbol{d}_i) = \sqrt{d_{i1}^2 + d_{i2}^2 + d_{i3}^2}, \quad \boldsymbol{s}_{U_i}(\boldsymbol{d}_i) = \frac{1}{h_{U_i}(\boldsymbol{d}_i)} \begin{bmatrix} d_{i1} & d_{i2} & d_{i3} \end{bmatrix}^T. \tag{4.35}$$

If $h_i > \mu_i d_{i1}$, on the other hand, $\boldsymbol{s}_{U_i}(\boldsymbol{d}_i)$ obtained by (4.35) is not in F_i and $h_{U_i}(\boldsymbol{d}_i)$ and $\boldsymbol{s}_{U_i}(\boldsymbol{d}_i)$ should be calculated by

$$h_{U_i}(\boldsymbol{d}_i) = \frac{d_{i1} + \mu_i h_i}{\sqrt{1 + \mu_i^2}}, \quad \boldsymbol{s}_{U_i}(\boldsymbol{d}_i) = \frac{1}{h_i \sqrt{1 + \mu_i^2}} \begin{bmatrix} h_i & \mu_i d_{i2} & \mu_i d_{i3} \end{bmatrix}^T. \tag{4.36}$$

The primitive contact force $\boldsymbol{s}_{U_i}(\boldsymbol{d}_i)$ obtained by (4.36) is on the boundary of F_i.

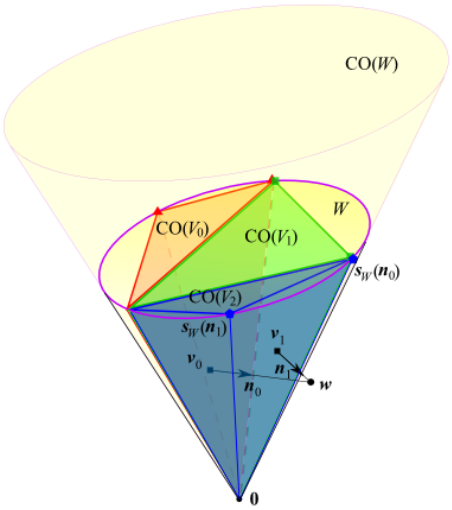


Figure 4.4. Illustration of the algorithm in 3-D space to compute the minimum distance between $\boldsymbol{w}$ and $V = \mathrm{CO}(W)$. By iteration it generates a sequence of simplicial cones $\mathrm{CO}(V_k)$ in the convex cone $\mathrm{CO}(W)$ such that the point $\boldsymbol{v}_k$ in $\mathrm{CO}(V_k)$ having the minimum Euclidean distance to the point $\boldsymbol{w}$ converges to the closest point in $\mathrm{CO}(W)$ to $\boldsymbol{w}$. Then, the distance between them gives the Euclidean separation distance between $\mathrm{CO}(W)$ and $\boldsymbol{w}$.

4.5.2 Computing Feasible Contact Forces

Here, I introduce an efficient algorithm to solve problem (4.23) and compute the minimum distance between $\boldsymbol{w}$ and V (Zheng and Chew, 2009). It generates a sequence of simplicial cones $\mathrm{CO}(V_k)$ in V such that their minimum distances to the point $\boldsymbol{w}$ converge to the minimum distance between $\boldsymbol{w}$ and V, as illustrated in Fig. 4.4. If $\boldsymbol{w} \notin V$, then the point $\hat{\boldsymbol{w}}$ in V closest to $\boldsymbol{w}$ in terms of the Euclidean distance will be computed as well, and it represents the closest wrench to the required one $\boldsymbol{w}$ that can be generated with feasible contact forces for tracking the reference motion.

Fig. 4.4 illustrates the iteration steps of the algorithm. Suppose that I have a linearly independent subset V_k of W such that $\boldsymbol{w} \notin \mathrm{CO}(V_k)$ but $\boldsymbol{w} \notin \mathrm{CO}(V_k)$. Let $d_k \triangleq \min_{\boldsymbol{v} \in \mathrm{CO}(V_k)} \|\boldsymbol{w} - \boldsymbol{v}\|$ be the minimum distance between $\boldsymbol{w}$ and $\mathrm{CO}(V_k)$ as defined by (4.23), $\boldsymbol{v}_k$ the point in $\mathrm{CO}(V_k)$ such that $d_k = \|\boldsymbol{w} - \boldsymbol{v}_k\|$, and $\hat{V}_k$ a minimal subset of V_k such that $\boldsymbol{v}_k$ can be written as a positive combination of $\hat{V}_k$. Since $\boldsymbol{w} \notin \mathrm{CO}(V_k)$, I have $\boldsymbol{v}_k \neq \boldsymbol{w}$. Let $\boldsymbol{n}_k$ be the unit vector from $\boldsymbol{v}_k$ to $\boldsymbol{w}$ and H_k the hyperplane with normal $\boldsymbol{n}_k$

passing through $\boldsymbol{v}_k$. Then, it turns out that H_k supports $\mathrm{CO}(V_k)$ such that $\boldsymbol{w}$ lies on the side of H_k that $\boldsymbol{n}_k$ points to and $\mathrm{CO}(V_k)$ falls on the other side.

The algorithm iterates by

$$V_{k+1} = \hat{V}_k \cup \{\boldsymbol{s}_W(\boldsymbol{n}_k)\} \tag{4.37}$$

where $\boldsymbol{s}_W(\boldsymbol{n}_k)$ is the support mapping of W along $\boldsymbol{n}_k$ as defined by (4.31). Since $\boldsymbol{v}_k \in \mathrm{CO}(V_k) \subset \mathrm{CO}(W)$, the hyperplane H_k cuts W such that $h_W(\boldsymbol{n}_k) \geq 0$ always holds. If $h_W(\boldsymbol{n}_k) > 0$, then $\boldsymbol{s}_W(\boldsymbol{n}_k)$ is on the same side of H_k as $\boldsymbol{w}$ and different side from $\mathrm{CO}(V_k)$. As a consequence, it can be proved that $d_{k+1} \triangleq \min_{\boldsymbol{v}\in\mathrm{CO}(V_{k+1})} \|\boldsymbol{w} - \boldsymbol{v}\|$, namely the minimum distance between $\boldsymbol{w}$ and $\mathrm{CO}(V_{k+1})$, is strictly smaller than d_k. Therefore, d_k is strictly decreasing along with the iteration and guaranteed to converge to the minimum distance between $\boldsymbol{w}$ and V, which is the greatest lower bound on d_k and can be obtained when $h_W(\boldsymbol{n}_k) = 0$. Thereby, the criterion for stopping the iteration is taken to be $h_W(\boldsymbol{n}_k) < \epsilon$. Along with the convergence of d_k, the point $\boldsymbol{v}_k$ converges to the point in V closest to $\boldsymbol{w}$, which I denote by $\hat{\boldsymbol{w}}$.

In addition to the minimum distance between $\boldsymbol{w}$ and V, the algorithm returns other important values so that I can compute the corresponding feasible contact forces to generate $\boldsymbol{w}$ if $\boldsymbol{w} \in V$ or $\hat{\boldsymbol{w}}$ if $\boldsymbol{w} \notin V$. Note that $\hat{V}_k$ is a linearly independent subset of W such that $\boldsymbol{v}_k$ can be written as its positive combination at every iteration. As $\boldsymbol{v}_k$ converges to $\hat{\boldsymbol{w}}$, $\hat{V}_k$ finally gives a linearly independent subset of W, denoted by $\hat{W}$, such that $\hat{\boldsymbol{w}}$ can be written as its positive combination

$$\hat{\boldsymbol{w}} = \sum_{j=1}^{n} c_j \boldsymbol{w}_j \text{ with } c_j > 0 \text{ for } \forall j. \tag{4.38}$$

From (4.37) it can be seen that every point in $\hat{W}$ is obtained by calculating the support mapping $\boldsymbol{s}_W$ of W along a certain direction. In the case where W is defined as (4.29a), from (4.32a) and (4.33) I obtain a contact index $i_j \in [1, m]$ and a primitive contact force $\boldsymbol{s}_j \in U_{i_j}$

for each $\boldsymbol{w}_j \in \hat{W}$ such that $\boldsymbol{w}_j = \boldsymbol{G}_{i_j}\boldsymbol{s}_j$, which means that $\boldsymbol{w}_j$ is generated by the primitive contact force $\boldsymbol{s}_j$ at contact i_j. Substituting it into (4.38), I then obtain

$$\hat{\boldsymbol{w}} = \sum_{j=1}^{n} c_j \boldsymbol{G}_{i_j}\boldsymbol{s}_j \text{ with } c_j > 0 \text{ for } \forall j. \tag{4.39}$$

As a result, I derive the contact forces $\boldsymbol{f}_i \in F_i$ $(i = 1, 2, \ldots, m)$ for generating $\hat{\boldsymbol{w}}$ as

$$\boldsymbol{f}_i = \sum_{\substack{j=1 \\ i_j = i}}^{n} c_j \boldsymbol{s}_j. \tag{4.40}$$

Since $\boldsymbol{s}_j$ is a primitive contact force and satisfies the friction constraint and all coefficients c_j are positive, $\boldsymbol{f}_i$ computed by 4.40 is guaranteed to satisfy the friction constraint.

If W is given by (4.29b) and σ by (4.25b), from (4.32b) and (4.33) I have a set of contact indices $\{i_{j,1}, i_{j,2}, \ldots, i_{j,m_j}\}$ and a set of primitive contact forces $\{\boldsymbol{s}_{j,1}, \boldsymbol{s}_{j,2}, \ldots, \boldsymbol{s}_{j,m_j}\}$ for each $\boldsymbol{w}_j \in \hat{W}$ such that $\boldsymbol{w}_j = \sum_{l=1}^{m_j} \boldsymbol{G}_{i_{j,l}}\boldsymbol{s}_{j,l}$, where $i_{j,l} \in [1, m]$ and $\boldsymbol{s}_{j,l} \in U_{i_{j,l}}$ for $l = 1, 2, \ldots, m_j$ and $m_j \leq m$. This means that $\boldsymbol{w}_j$ is the resultant wrench from primitive contact forces at m_j contacts. From (4.38), I then derive

$$\hat{\boldsymbol{w}} = \sum_{j=1}^{n} \left(c_j \sum_{l=1}^{m_j} \boldsymbol{G}_{i_{j,l}}\boldsymbol{s}_{j,l} \right) \text{ with } c_j > 0 \text{ for } \forall j. \tag{4.41}$$

Therefore, I obtain another set of feasible contact forces to generate $\hat{\boldsymbol{w}}$ as

$$\boldsymbol{f}_i = \sum_{\substack{j=1,\, l=1 \\ i_{j,l} \,=\, i}}^{n,\, m_j} c_j \boldsymbol{s}_{j,l}. \tag{4.42}$$

The pseudocode of this algorithm is provided in **Algorithm 1**.

Algorithm 1 Algorithm for feasible contact forces

Input: $\boldsymbol{w}$ and W
Output: Feasible total contact force $\boldsymbol{f}$

$V_0 \leftarrow$ any linearly independent subset of W, $k \leftarrow 0$
$\boldsymbol{v}_0 \leftarrow$ point in $\mathrm{CO}(V_0)$ whose distance from $\boldsymbol{w}$ is the minimum distance between $\boldsymbol{w}$ and $\mathrm{CO}(V_0)$
$\hat{V}_0 \leftarrow$ minimal subset of V_0 such that $\boldsymbol{v}_0 \in \mathrm{CO}(\hat{V}_0)$
$\boldsymbol{n}_0 \leftarrow -\boldsymbol{v}_0/\|\boldsymbol{v}_0\|_2$
while $h_W(\boldsymbol{n}_k) > \epsilon$ **do**
  $V_{k+1} \leftarrow \hat{V}_k \cup \{\boldsymbol{s}_W(\boldsymbol{n}_k)\}$, $k \leftarrow k+1$
  $\boldsymbol{v}_k \leftarrow$ point in $\mathrm{CO}(V_k)$ whose distance from $\boldsymbol{w}$ is the minimum distance between $\boldsymbol{w}$ and $\mathrm{CO}(V_k)$
  $\hat{V}_k \leftarrow$ minimal subset of V_k such that $\boldsymbol{v}_k \in \mathrm{CO}(\hat{V}_k)$
  $\boldsymbol{n}_k \leftarrow -\boldsymbol{v}_k/\|\boldsymbol{v}_k\|_2$
end while
Compute $\boldsymbol{f}_i$ by (4.40) or (4.42)
return $\boldsymbol{f}$

4.5.3 Computing Minimum Contact Forces

From (4.40) and (4.42) it can be seen that the solution for feasible contact forces to generate $\hat{\boldsymbol{w}}$ may not be unique. It is desired to have small contact forces to protect the robot from overload and damage. This may also help reduce the resulting joint torques. Here I discuss algorithms to compute the minimum feasible contact forces (Zheng et al., 2012; Zheng and Yamane, 2013b).

In the following discussion, let $\boldsymbol{z}$ be the farthest point in W from the origin in the direction $\hat{\boldsymbol{w}}$, as depicted in Fig. 4.5. Then, $\boldsymbol{z}$ represents the largest resultant wrench in the direction $\hat{\boldsymbol{w}}$ that can be generated by a feasible total contact force with unit overall magnitude. This implies that the minimum overall magnitude σ of a feasible total contact force $\boldsymbol{f}$ to generate $\hat{\boldsymbol{w}}$ is equal to $\|\hat{\boldsymbol{w}}\|/\|\boldsymbol{z}\|$. In what follows, I continue with the result of **Algorithm 1** to compute $\boldsymbol{z}$ and the minimum contact forces.

From **Algorithm 1**, I obtain a linearly independent subset $\hat{W}$ of W such that $\hat{\boldsymbol{w}}$ can be written as its positive combination (4.38). Let $\sigma_{k=0} \triangleq \sum_{j=1}^{n} c_j$ and $\boldsymbol{z}_{k=0} \triangleq \hat{\boldsymbol{w}}/\sigma_0$. From (4.38) it follows that $\boldsymbol{z}_0$ is a convex combination of $\hat{W}$, which implies that $\boldsymbol{z}_0 \in \mathrm{CH}(\hat{W})$.

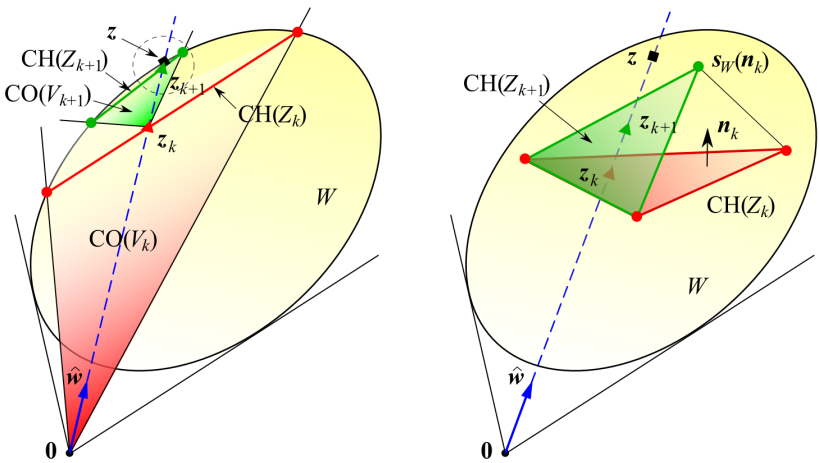


Figure 4.5. Illustration of two iteration strategies to compute the minimum contact forces. (a) A subset V_{k+1} of $W - \boldsymbol{z}_k$ is computed at every iteration by **Algorithm 1** such that $\hat{\boldsymbol{w}}$ is a positive combination of V_{k+1}. Then, $\boldsymbol{z}_{k+1}$, a convex combination of $Z_{k+1} = V_{k+1} + \boldsymbol{z}_k$ that is proportional to $\hat{\boldsymbol{w}}$, is closer to $\boldsymbol{z}$ than $\boldsymbol{z}_k$. Hence, $\boldsymbol{z}_k$ approaches $\boldsymbol{z}$ as the iteration proceeds. (b) $\mathrm{CH}(Z)_k$ (red triangle) is a facet in W such that its interior contains a point $\boldsymbol{z}_k$ proportional to $\hat{\boldsymbol{w}}$ and $\boldsymbol{n}_k$ is the normal of the facet such that $\boldsymbol{n}_k^T \hat{\boldsymbol{w}} > 0$. Using a subset of the vertices of $\mathrm{CH}(Z)_k$ and $\boldsymbol{s}_W(\boldsymbol{n}_k)$, I can find a new facet (green triangle) that contains a new point $\boldsymbol{z}_{k+1}$ in the direction $\hat{\boldsymbol{w}}$ strictly closer to $\boldsymbol{z}$ than $\boldsymbol{b}_k$. This iteration can proceed until $\boldsymbol{z}_k$ converges to $\boldsymbol{z}$, provided that $\boldsymbol{z}_{k+1}$ always lies in the interior of the new facet.

Then, $\boldsymbol{z}_0 \in W$ since $\mathrm{CH}(\hat{W}) \subset W$. However, $\boldsymbol{z}_0$ may not be the farthest point $\boldsymbol{z}$ in W from the origin in the direction $\hat{\boldsymbol{w}}$, so the contact forces computed by (4.40) or (4.42) may not be minimal. To obtain the farthest point $\boldsymbol{z}$, the general idea is to iteratively push the point $\boldsymbol{z}_k$ further in the direction $\hat{\boldsymbol{w}}$ until it converges to $\boldsymbol{z}$. I discuss two iteration strategies to do so as follows.

Strategy 1: This iteration strategy (Zheng and Yamane, 2013b) is illustrated in Fig. 4.5a. Let $W - \boldsymbol{z}_k$ be the set obtained from W by shifting the origin to $\boldsymbol{z}_k$. Since $\hat{\boldsymbol{w}}$ is contained in $\mathrm{CO}(W)$ and $\boldsymbol{z}_k$ is a point in W, $\hat{\boldsymbol{w}}$ is also contained in $\mathrm{CO}(W - \boldsymbol{z}_k)$. Then, using **Algorithm 1** again to compute the minimum distance between $\hat{\boldsymbol{w}}$ and $\mathrm{CO}(W - \boldsymbol{z}_k)$, I obtain a linearly independent set $\hat{V}_{k+1}$ of points in $W - \boldsymbol{z}_k$ such that $\hat{\boldsymbol{w}}$ can be written as their positive combination. Let $Z_{k+1} = \hat{V}_{k+1} + \boldsymbol{z}_k$, which is a subset of W, and $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_L$ denote the points in Z_{k+1}. Then, $\hat{V}_{k+1} = \{\boldsymbol{w}_1 - \boldsymbol{z}_k, \boldsymbol{w}_2 - \boldsymbol{z}_k, \ldots, \boldsymbol{w}_L - \boldsymbol{z}_k\}$ and $\hat{\boldsymbol{w}}$ can be

written as

$$\hat{\boldsymbol{w}} = \sum_{l=1}^{L} c_l(\boldsymbol{w}_l - \boldsymbol{z}_k) \text{ with } c_l > 0 \text{ for } \forall l. \tag{4.43}$$

Let $\sigma_{k+1} = \sum_{l=1}^{L} c_l$ and

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \frac{1}{\sigma_{k+1}}\hat{\boldsymbol{w}} = \sum_{l=1}^{L} \frac{c_l}{\sigma_{k+1}}\boldsymbol{w}_l. \tag{4.44}$$

Equation (4.44) shows that $\boldsymbol{z}_{k+1}$ is proportional to $\hat{\boldsymbol{w}}$ and it is also a convex combination of Z_{k+1}, as depicted in Fig. 4.5a. Since Z_{k+1} is a subset of W, I have $\mathrm{CH}(Z_{k+1}) \subset W$ and $\boldsymbol{z}_{k+1} \in W$. Moreover, since $\sigma_{k+1} > 0$, $\boldsymbol{z}_{k+1}$ given by (4.44) is strictly further from the origin and close to $\boldsymbol{z}$ than $\boldsymbol{z}_k$, which guarantees that $\boldsymbol{z}_k$ will converge to $\boldsymbol{z}$ as the iteration proceeds.

When $\boldsymbol{z}_k$ is close enough to $\boldsymbol{z}$ and to the boundary of W at one iteration, **Algorithm 1** terminates with $h_{W-\boldsymbol{z}_k}(\boldsymbol{n}) < \epsilon_{\mathrm{ZC}}$ and the minimum distance between $\hat{\boldsymbol{w}}$ and $\mathrm{CO}(A - \boldsymbol{z}_k)$ being nonzero, where $\boldsymbol{n}$ is a unit normal vector occurring at an iteration of **Algorithm 1**. Since $h_{W-\boldsymbol{z}_k}(\boldsymbol{n}) = h_W(\boldsymbol{n}) - \boldsymbol{n}^T\boldsymbol{z}_k$ and $\boldsymbol{n}^T\boldsymbol{z} \leq h_W(\boldsymbol{n})$, I obtain $\boldsymbol{n}^T(\boldsymbol{z} - \boldsymbol{z}_k) < \epsilon$, or equivalently $\|\boldsymbol{z} - \boldsymbol{z}_k\| < \epsilon_{\mathrm{ZC}}\|\hat{\boldsymbol{w}}\|/\boldsymbol{n}^T\hat{\boldsymbol{w}}$. Hence, I can use $\boldsymbol{z}_k$ to approximate $\boldsymbol{z}$. Furthermore, the set Z_k finally offers a subset of W, denoted by Z, such that $\boldsymbol{z}$ can be written as its convex combination. Since $\boldsymbol{z}$ and $\hat{\boldsymbol{w}}$ are in the same direction, $\hat{\boldsymbol{w}}$ can be written as a positive combination of Z. Similarly to the set $\hat{W}$ computed by **Algorithm 1**, every element of the Z is obtained by calculating the support mapping of W in a certain direction. Therefore, I also attain a set of corresponding primitive contact forces to generate the elements of Z. Then, substituting these primitive contact forces into (4.40) or (4.42) for those generating $\hat{W}$, I obtain the feasible contact forces with minimal overall magnitude to generate $\hat{\boldsymbol{w}}$.

Strategy 2: This iteration strategy (Zheng et al., 2010, 2012) is illustrated in Fig. 4.5b. During the above iteration, the set Z_k can consist of six linearly independent points in W in $\mathbb{R}^6$ and $\boldsymbol{z}_k$ is proportional to $\hat{\boldsymbol{w}}$ and in the interior of the convex hull $\mathrm{CH}(Z_k)$ of Z_k, as depicted in Fig. 4.5b. Then, $\mathrm{CH}(Z_k)$ is a facet contained in W. Let $\boldsymbol{n}_k$ be the normal of

facet $\mathrm{CH}(Z_k)$ that satisfies $\boldsymbol{n}_k^T \hat{\boldsymbol{w}} > 0$. Since W is convex and facet $\mathrm{CH}(Z_k)$ is contained in W, $h_W(\boldsymbol{n}_k) \geq \boldsymbol{n}_k^T \boldsymbol{z}_k$ always holds.

If $h_W(\boldsymbol{n}_k) > \boldsymbol{n}_k^T \boldsymbol{z}_k$, then the point $\boldsymbol{s}_W(\boldsymbol{n}_k)$ lies on the different side of facet $\mathrm{CH}(Z_k)$ from the origin. Furthermore, $\boldsymbol{s}_W(\boldsymbol{n}_k)$ and Z_k are affinely independent. Since $\boldsymbol{s}_W(\boldsymbol{n}_k)$ and any five points in Z_k form a new facet in W, I have six new facets in W in total. Among them, at least one facet intersects the ray in the direction $\hat{\boldsymbol{w}}$ originating from the origin. Since $\boldsymbol{z}_k$ is in the interior of $\mathrm{CH}(Z_k)$ and $h_W(\boldsymbol{n}_k) > \boldsymbol{n}_k^T \boldsymbol{z}_k$, it can be proven that their intersection point is strictly farther from the origin and closer to the point $\boldsymbol{z}$ than $\boldsymbol{z}_k$. I use this intersection point as $\boldsymbol{z}_{k+1}$. If $\boldsymbol{z}_{k+1}$ is in the interior of the new facet, then I can construct Z_{k+1} using the vertices of the facet and repeat this iteration.

If $h_W(\boldsymbol{n}_k) = \boldsymbol{n}_k^T \boldsymbol{z}_k$ on the other hand, then the hyperplane with normal $\boldsymbol{n}_k$ passing through $\boldsymbol{z}_k$ supports W at $\boldsymbol{z}_k$, which implies that $\boldsymbol{z}_k$ is the farthest point in W from the origin in the direction $\hat{\boldsymbol{w}}$. Therefore, I can take $\boldsymbol{z}$ to be $\boldsymbol{z}_k$ and stop the iteration. At the same time, Z_k gives the set Z, which consists of linearly independent points in W to represent $\boldsymbol{z}$ as its positive combination. Hence, the condition $h_W(\boldsymbol{n}_k) - \boldsymbol{n}_k^T \boldsymbol{z}_k < \epsilon$ can be used as the stopping criterion, and from it I derive $\|\boldsymbol{z} - \boldsymbol{z}_k\| < \epsilon \|\hat{\boldsymbol{w}}\| / \boldsymbol{n}_k^T \hat{\boldsymbol{w}}$.

Hybrid Use: Both iteration strategies compute $\boldsymbol{z}$ by generating a sequence of points $\boldsymbol{z}_k$ in A between the origin and $\boldsymbol{z}$ that eventually converge to $\boldsymbol{z}$. Nevertheless, the computation cost for an iteration of the second strategy is much smaller. To determine if a facet contains a point on the ray along $\hat{\boldsymbol{w}}$ and compute this point as $\boldsymbol{z}_{k+1}$, I only need to solve a system of six linear equations. By contrast, the first iteration strategy needs to call **Algorithm 1**, which will take several iterations, to compute $\boldsymbol{z}_{k+1}$. Although $\boldsymbol{b}_{k+1}$ obtained by the first strategy could be closer to $\boldsymbol{z}$, the overall cost for computing $\boldsymbol{z}$ can be reduced by using the second one. However, it should be pointed out that the second iteration strategy can proceed only if $\boldsymbol{b}_k$ is in the interior of the facet, which is not guaranteed. It is possible that $\boldsymbol{b}_k$ is on the boundary of the facet at a certain iteration. In that case, I can simply switch to the first iteration strategy to compute $\boldsymbol{z}_{k+1}$. By such a hybrid use of the two iteration strategies,

I have an algorithm with both high efficiency and ensured convergence to compute the minimum contact forces.

4.6 Simulation

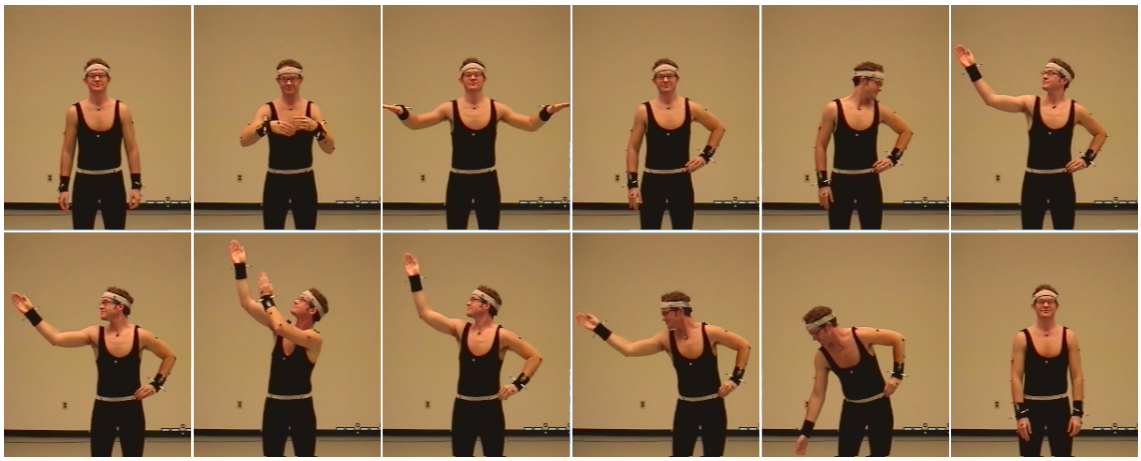
4.6.1 Simulation Setup

I use the dynamics simulator with rigid-contact model developed by University of the Tokyo (Yamane and Nakamura, 2008a,b) to conduct the experiments. The humanoid robot model used in the simulations has 25 joints and 31 DOFs including the translation and rotation of the floating base. Each leg has 7 joints (pitch, roll, yaw at both the hip and the ankle and pitch at the knee). I only consider 4 joints in each arm (pitch, roll, yaw at the shoulder and pitch at the elbow) and fix wrist joints. There are 3 joints in the torso. The robot model is about 1.7 meters tall and 65 kg in weight.

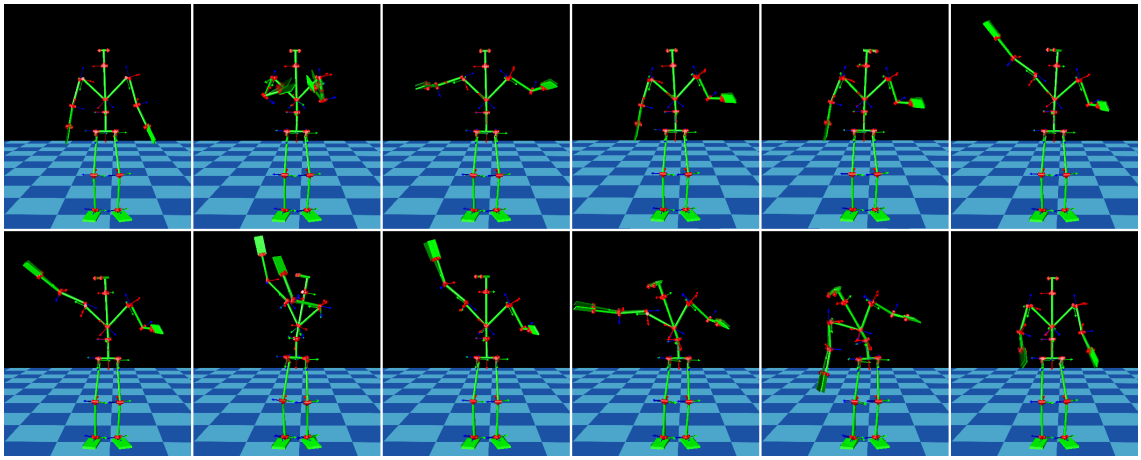
I implement the controller in C++ on a laptop with an Intel Core i7 2.67GHz CPU and 3GB RAM. **Algorithm 1** is used to compute the contact forces in the controller. The average computation time of the whole controller in the following examples is in the range of 1.48–1.61 msec, which is fast enough for realtime control at 500 Hz. Contact force and joint torque optimizations take approximately 24–26% and 31–33% of the time respectively. The rest is spent for computing the other quantities such as the desired accelerations, mass matrix, and Jacobian matrices.

4.6.2 Tracking Human Motion Without Contact State Change

In this example, the robot tracks two motion capture clips chosen from CMU Motion Capture Data Library (`http://mocap.cs.cmu.edu/`), where two actors perform nursery rhyme "I'm a little teapot" while maintaining their feet on the ground, as displayed in Figs. 4.6a and 4.7a. The simulated motions are shown in Figs. 4.6b and 4.7b and a video on `http://www.cs.unc.edu/~yuzheng/dissertation/`, which demonstrates that the proposed tracking controller enables the robot to reproduce the human motions



(a)

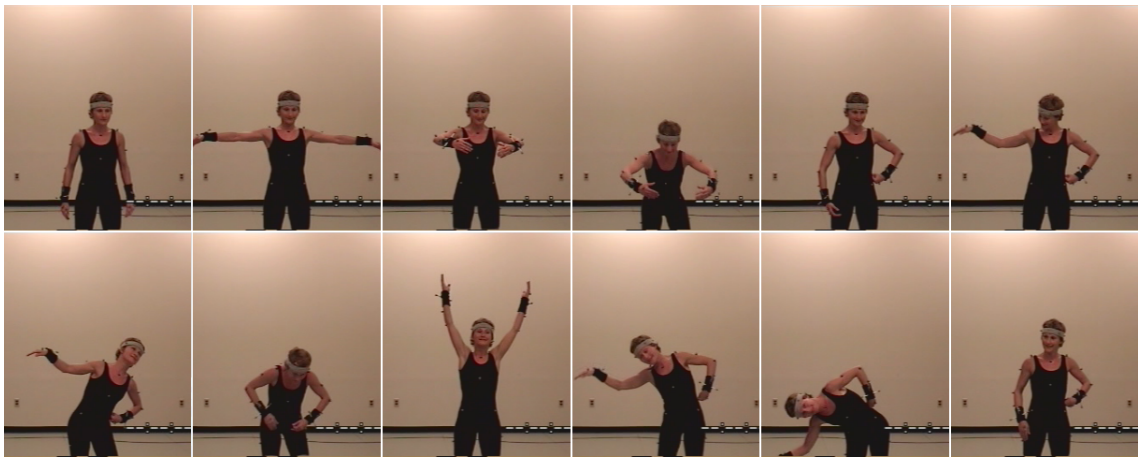


(b)

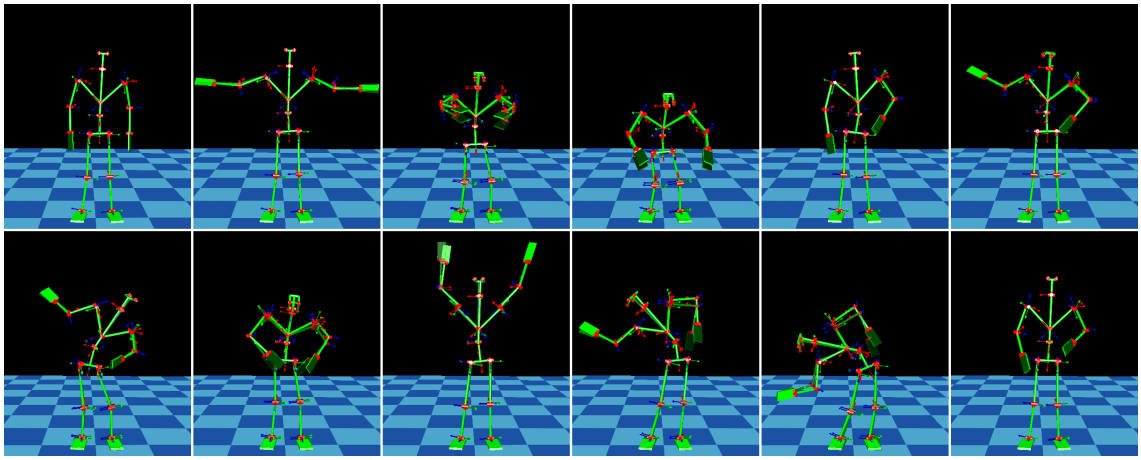
Figure 4.6. Example of (a) original and (b) simulated motion of "I'm a little teapot" performed by subject 1.

without falling. For the second case (Fig. 4.7), I also plot the contact forces and moments on the left foot computed by the tracking controller and the simulated values in Fig. 4.8. It can be observed that the two values match well in all six components. While not shown, I see similar match at the right foot.

I do note the difference in tracking fidelity between the two examples due to the different styles of the motions. Because the first example is slower and smoother, the robot can track the motion more accurately. In the second example, the robot's pose can be significantly different from the subject's, especially when the subject makes rapid movements.



(a)



(b)

Figure 4.7. Example of (a) original and (b) simulated motion of "I'm a little teapot" performed by subject 2.

4.6.3 Tracking Human Motion With Contact State Change

In the second example I let the robot follow human's stepping motions. The simulated robot motions are shown in Figs. 4.9 and 4.10 and a video on `http://www.cs.unc.edu/~yuzheng/dissertation/`. By the proposed tracking controller, the robot can preserve the style of original human motions. As the side stepping motion is performed in the xz plane of the global frame, the contact forces in x- and z-directions and the moment in y-direction on the feet play a decisive role in the motion, and the optimized and actual values on both feet are exhibited in Fig. 4.11.

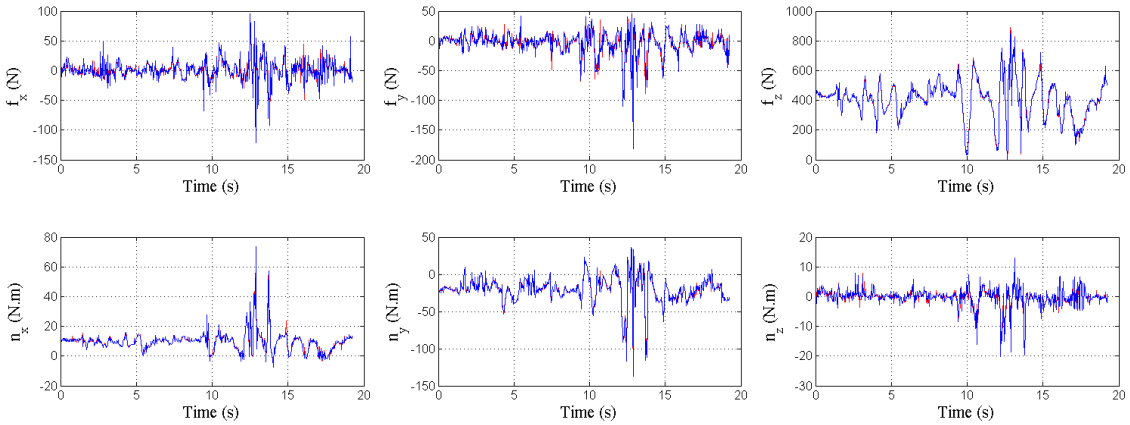


Figure 4.8. Optimized (red) and actual (blue) contact forces and moments on the left foot.

4.6.4 Tracking Extreme Reference Motion

Only considering the contact force constraints does not generally guarantee that the robot does not tip over, especially if the reference motion is extremely difficult to track. A simple example is trying to maintain a stationary pose where the COM projection is outside of the contact area. In this case, the tracking controller will result in a falling motion with COP in the contact area. This issue is not unique to my controller but applies to any realtime controller for interactive floating-base robots because I cannot predict future reference motion.

A solution to this issue is to switch or interpolate between two or more reference motions, one being a "safe" reference such as maintaining a static equilibrium pose. Similar idea has been used in an online walking pattern generator that uses joystick input to determine the walking direction (Nishiwaki et al., 2002).

I demonstrate this concept with a simple example using the proposed tracking controller. The main reference motion is a static pose that is not a static equilibrium for robot's mass distribution, as depicted by the transparent model in Fig. 4.12a. Tracking this motion will eventually cause the robot to fall. I therefore use another reference motion, which is maintaining a static equilibrium pose shown by the opaque model in Fig. 4.12a. It is also the initial pose for the simulation.

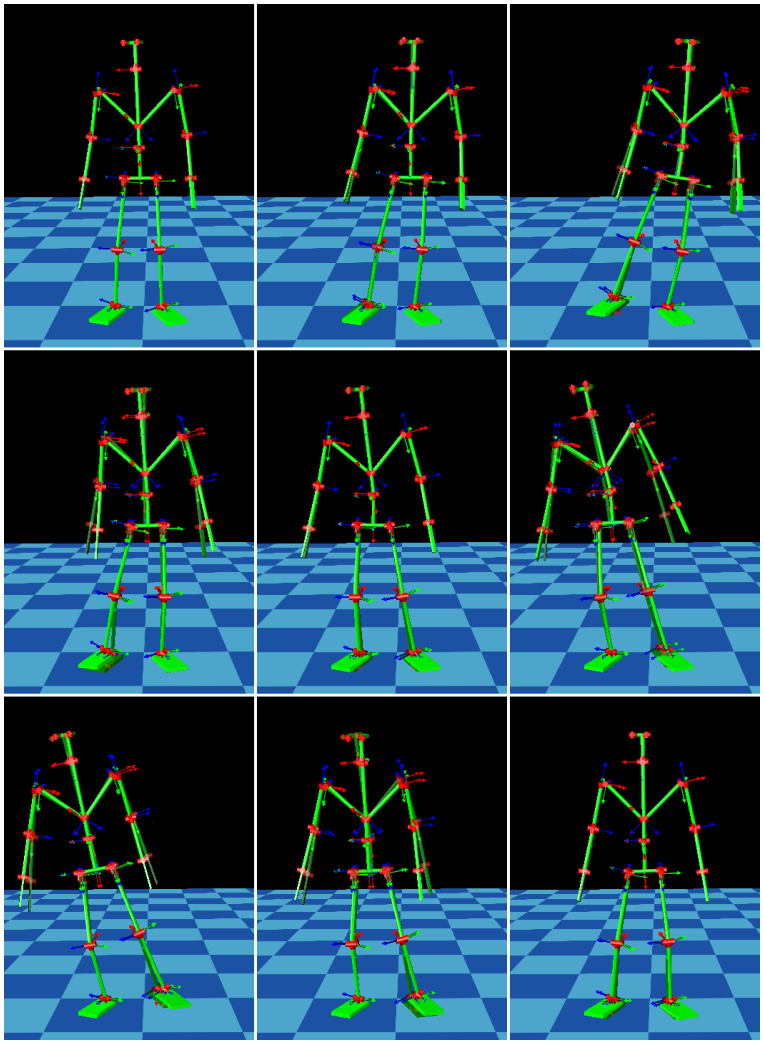


Figure 4.9. Simulated side stepping motion.

In this example, I interpolate the two motions with weights determined by the time at which the COM is expected to reach the support area boundary computed based on the current position and velocity of the COM. The shorter the time is, the larger the weight for the static equilibrium motion. If the time is below a threshold, I completely switch the reference pose to the static equilibrium one to prevent the robot from falling. In the simulation, the robot finally reaches and maintains the pose shown in Fig. 4.12b, which is between the two reference poses. Figure 4.12c shows that the final COP is at the edge of the support area (toe).

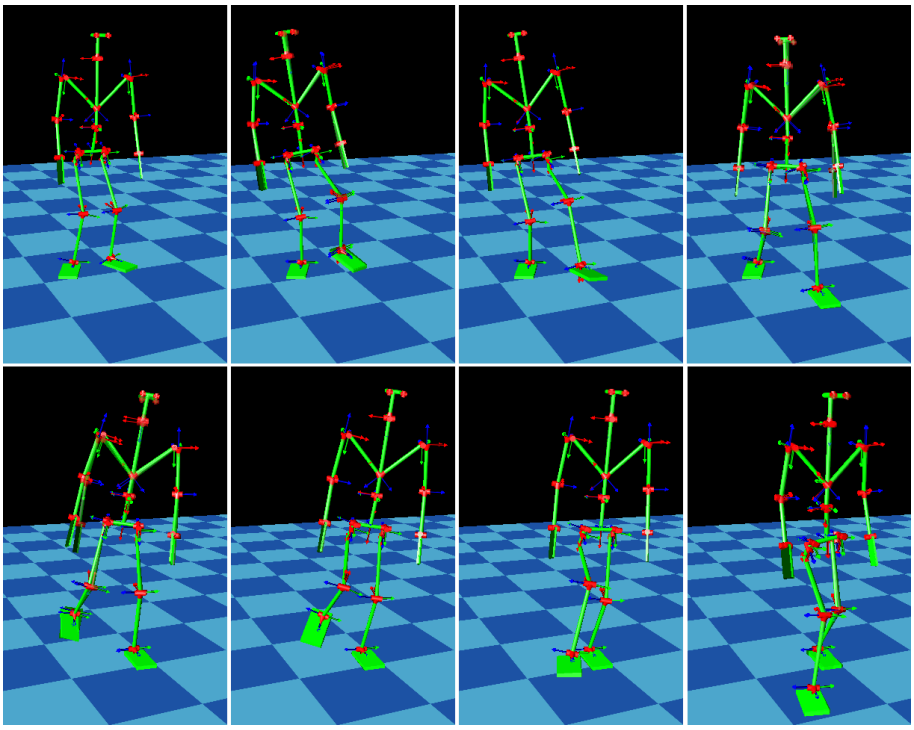


Figure 4.10. Simulated forward stepping motion.

4.7 Conclusions and Future Work

In this chapter, I present a controller for humanoid robots to track motion capture data. Given the desired accelerations at all DOF to track the reference motion, the proposed tracking controller computes the optimal joint torques and contact forces to realize the desired accelerations considering the full-body dynamics of the robot and the constraints on the contact forces. Simulation results show that the tracking controller successfully makes a humanoid robot track various human motions. I also illustrate a simple extension for preventing the robot from falling when an extreme motion is given as the reference.

This work can be extended in many directions. A straightforward extension is to the motions involving contacts at hands or other links in addition to the feet, which will allow humanoid robots to track a much larger range of human motions. Another possible extension is to the motions with more complex contact states and contact link motions. In addition to the rolling contact formulated in Section 4.4.1, I can extend the formulation to the sliding

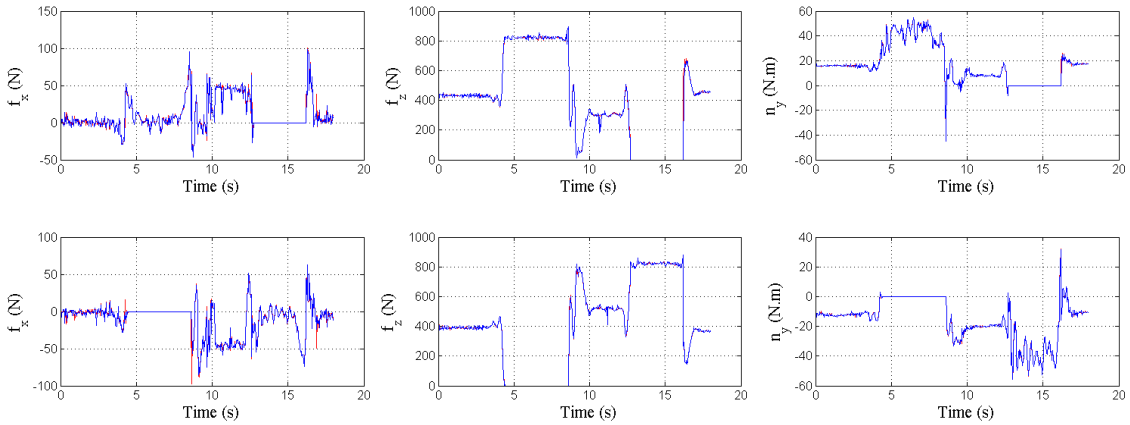
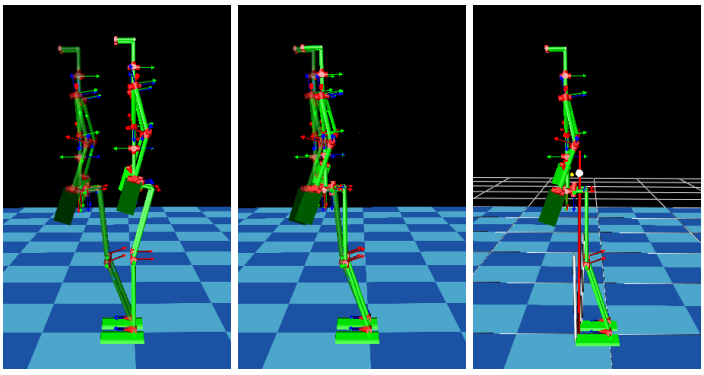


Figure 4.11. Optimized (red) and actual (blue) contact forces in x- and z-directions and moments in y-direction on the left (upper) and right (below) feet for tracking the stepping motion.

contact, where the contact force should be restricted to the boundary of the friction cone and opposite to the sliding direction.



(a) (b) (c)

Figure 4.12. Example of preventing falling in tracking extreme motions. (a) Unbalanced reference pose for tracking (transparent) and static equilibrium pose for falling prevention (opaque). (b) Final pose of the robot, which is close to but does not reach the tracking reference pose. (c) The contact forces at the local COP on each foot (white lines) and the total contact force at the global COP (red line), which reaches the boundary of the support area.

CHAPTER 5: CONCLUSION

One day in the future humanoid robots are expected to work with humans and provide assistance and services to people. Prior to such applications, humanoid robots need to possess some basic locomotion abilities, such as balancing, walking and running. To make them fully competent to work in a complex human environment, a humanoid robot should also be able to manipulate the environment and perform dynamic locomotion tasks in addition to those basic motions. Moreover, since humanoid robots are supposed to work in human society, they are desired to behave like a human while maintaining balance in order to make themselves human-friendly. Some tasks, such as dancing and orchestra conducting, also require stylized human-like motions. Considering these more advanced locomotion tasks in addition to the basic ones is an essential step to make humanoid robots truly useful.

As an example of manipulating the environment and performing dynamic motions, I investigate how to let a humanoid robot balance and walk on a cylinder that can roll freely on the horizontal ground. Based on a simplified model that includes the robot and the cylinder, I design a balance controller that enables the robot to maintain balance on the cylinder under external disturbances. The effectiveness of the controller has been verified with simulation and hardware experiments. I also develop the methods for the robot to generate walking behaviors on the cylinder and roll the cylinder at a desired speed.

To endow a humanoid robot with human-like motions, I propose a motion tracking controller for the robot to imitate captured human motions while maintaining balance, considering the exact full-body dynamics of the robot and the strict constraint on the contact forces with the environment. The motion tracking controller consists of a PD controller that generates the desired joint accelerations for motion tracking and an optimization problem that computes the joint torques and contact forces for realizing the desired joint accelerations.

By taking advantage of the property that the joint torques do not contribute to the six DOF of the floating base, I decouple the computation of contact forces and joint torques into two sequential steps such that the optimization with strict contact force constraints is solved in real time. By full-body simulation it is shown that a humanoid robot can reproduce various human motions without a fall by using this motion tracking controller.

Through the work in this dissertation, I propose that, by considering dynamic and kinematic constraints in the environment in the controller design, humanoid robots can achieve more complex locomotion tasks by manipulating a dynamic object or tracking given reference motions, while maintaining balance.

BIBLIOGRAPHY

Anderson, S. O. and Hodgins, J. K. (2010). Adaptive torque-based control of a humanoid robot on an unstable platform. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, pages 511–517, Nashville, TN.

Ayaz, Y., Owa, T., Tsujita, T., Konno, A., Munawar, K., and Uchiyama, M. (2009). Footstep planning for humanoid robots among obstacles of various types. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, pages 361–366, Paris, France.

Billard, A., Calinon, S., Dillman, R., and Schaal, S. (2008). Robot programming by demonstration. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, pages 1371–1394. Springer.

Boutin, L., Eon, A., Zeghloul, S., and Lacouture, P. (2010). An auto-adaptable algorithm to generate human-like locomotion for different humanoid robots based on motion capture data. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 634–639, Taipei, Taiwan.

Chestnutt, J., Lau, M., Cheung, G., Kuffner, J., Hodgins, J., and Kanade, T. (2005). Footstep planning for the honda asimo humanoid. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 629–634, Barcelona, Spain.

Collins, S. H., Ruina, A., Tedrake, R., and Wisse, M. (2005). Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085.

Collins, S. H., Wisse, M., and Ruina, A. (2001). A three-dimensional passive-dynamic walking robot with two legs and knees. *Int. J. Robot. Res.*, 20(7):607–615.

da Silva, M., Abe, Y., and Popović, J. (2008). Interactive simulation of stylized human locomotion. *ACM Trans. Graphics*, 27(3).

Dertien, E. (2006). Dynamic walking with dribbel. *IEEE Robot. Automat. Mag.*, 13(3):118–121.

Endo, T. and Nakamura, Y. (2005). An omnidirectional vehicle on a basketball. In *Proc. Int. Conf. Advanced Robot.*, pages 573–578.

Franken, M., van Oort, G., and Stramigioli, S. (2008). Analysis and simulation of fully actuated planar biped robots. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 634–639, Nice, France.

Freidovich, L. B., Mettin, U., Shiriaev, A. S., and Spong, M. W. (2009). A passive 2-DOF walker: hunting for gaits using virtual holonomic constraints. *IEEE Trans. Robot.*, 25(5):1202–1208.

Garcia, M., Chatterjee, A., Ruina, A., and Coleman, M. (1998). The simplest walking model: stability, complexity, and scaling. *ASME J. of Biomech. Eng.*, 120:281–288.

Goswami, A., Thuilot, B., and Espiau, B. (1998). A study of the passive gait of a compass-like biped robot: symmetry and chaos. *Int. J. Robot. Res.*, 17(12):1282–1301.

Harada, Y., Takahashi, J., Nenchev, D., and Sato, D. (2010). Limit cycle based walk of a powered 7DOF 3D biped with flat feet. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3623–3628, Taipei, Taiwan.

Hirukawa, H., Hattori, S., Harada, K., Kajita, S., Kaneko, K., Kanehiro, F., Fujiwara, K., and Morisawa, M. (2006). A universal stability criterion of the foot contact of legged robots - adios zmp. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1976–1983, Orlando, Florida.

Hobbelen, D. G. E. and Wisse, M. (2008a). Ankle actuation for limit cycle walkers. *Int. J. Robot. Res.*, 27(6):709–735.

Hobbelen, D. G. E. and Wisse, M. (2008b). Controlling the walking speed in limit cycle walking. *Int. J. Robot. Res.*, 27(9):989–1005.

Hyon, S.-H. (2009). Compliant terrain adaptation for biped humanoids without measuring ground surface and contact forces. *IEEE Transactions on Robotics*, 25(1):171–178.

Ikemata, Y., Akihito, S., and Fujimoto, H. (2003). Analysis of limit cycle in passive walking. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 601–606, Las Vegas, Nevada.

Ikemata, Y., Yasuhara, K., Sano, A., and Fujimoto, H. (1999). Making feasible walking motion of humanoid robots from human motion captured data. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1044–1049, Detroit, MI.

Ikemata, Y., Yasuhara, K., Sano, A., and Fujimoto, H. (2008). Generation and local stabilization of fixed point based on a stability mechanism of passive walking. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1588–1593, Pasadena, CA.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1620–1626, Taipei, Taiwan.

Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. (2001). The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 239–246, Maui, Hawaii.

Kajita, S., Nagasaki, T., Kaneko, K., Yokoi, K., and Tanie, K. (2005). A running controller of humanoid biped HRP-2LR. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 616–622, Barcelona, Spain.

Kajita, S. and Tanie, K. (1995). Experimental study of biped dynamic walking in the linear inverted pendulum mode. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 2885–2891, Nagoya, Japan.

Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. (2001). Footstep planning among obstacles for biped robots. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 500–505, Maui, HI.

Kumagai, M. and Ochiai, T. (2009). Development of a robot balancing on a ball – application of passive motion to transport. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 4106–4111, Kobe, Japan.

Kuo, A. D. (2002). Energetics of actively powered locomotion using the simplest walking model. *ASME J. of Biomech. Eng.*, 124(2):113–120.

Kuroki, Y., Fujita, M., Ishida, T., Nagasaka, K., and Yamaguchi, J. (2003). A small biped entertainment robot exploring attractive applications. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 471–476, Taipei, Taiwan.

Kuroki, Y., Kato, K., Nagasaka, K., Miyamoto, A., Ueno, K., and Yamaguchi, J. (2004). Motion evaluating system for a small biped entertainment robot. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 3809–3814, New Orieans, LA.

Lauwers, T. B., Kantor, G. A., and Hollis, R. L. (2006). A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 2884–2889, Orlando, Florida.

McGeer, T. (1990). Passive dynamic walking. *Int. J. Robot. Res.*, 9(2):62–82.

Miura, K., Morisawa, M., Kanehiro, F., Kajita, S., Kaneko, K., and Yokoi, K. (2011). Human-like walking with toe supporting for humanoids. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4428–4435, San Francisco, CA.

Miura, K., Morisawa, M., Nakaoka, S., Kanehiro, F., Harada, K., Kaneko, K., and Kajita, S. (2009). Robot motion remix based on motion capture data – towards human-like locomotion of humanoid robots. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, pages 596–603, Paris, France.

Muico, U., Lee, Y., Popović, J., and Popović, Z. (2009). Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graphics*, 28(3).

Nagarajan, U., Kantor, G. A., and Hollis, R. L. (2013). The ballbot: An omnidirectional balancing mobile robot. *Int. J. Robot. Res.*, in press.

Nagarajan, U., Mampetta, A., Kantor, G. A., and Hollis, R. L. (2009). State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 998–1003, Kobe, Japan.

Nakamura, Y. and Hanafusa, H. (1986). Inverse kinematics solutions with singulairty robustness for robot manipulator control. *ASME J. Dyn. Syst. Meas. Control*, 108(3):163–171.

Nakaoka, S., Nakazawa, A., Yokoi, K., Hirukawa, H., and Ikeuchi, K. (2003). Generating whole body motions for a biped robot from captured human dances. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 3905–3910, Taipei, Taiwan.

Nishiwaki, K., Kagami, S., Kuniyoshi, Y., Inaba, M., and Inoue, H. (2002). Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 2684–2689.

Osuka, K. and Kirihara, K. (2000). Motion analysis and experiments of passive walking robot QUARTET II. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 3052–3056, San Francisco, CA.

Ott, C., Lee, D., and Nakamura, Y. (2008). Motion capture based human motion recognition and imitation by direct marker control. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, pages 399–405, Daejeon, Korea.

Ott, C., Roa, M. A., and Hirzinger, G. (2011). Posture and balance control for biped robots based on contact force optimization. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, pages 26–33, Bled, Slovenia.

Safonova, A., Hodgins, J., and Pollard, N. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graphics*, 23(3):514–521.

Safonova, A., Pollard, N., and Hodgins, J. (2003). Optimizing human motion for the control of a humanoid robot. In *Int. Symp. Adaptive Motion of Animals and Machines*.

Schaal, S., Ijspeert, A., and Billard, A. (2003). Computational approaches to motor learning by imitation. *Phylosophical Transactions of the Royal Society of London B: Biological Sciences*, 358:537–547.

Sok, K., Kim, M., and Lee, J. (2007). Simulating biped behaviors from human motion data. *ACM Trans. Graphics*, 26(3).

Sugihara, T. (2008). Simulated regulator to synthesize ZMP manipulation and foot location for autonomous control of biped robots. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1264–1269, Pasadena, CA.

Sugihara, T., Nakamura, Y., and Inoue, H. (2002). Realtime humanoid motion generation through ZMP manipulation based on inverted pendulum control. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1404–1409, Washington, DC.

Tak, S., Song, O., and Ko, H. (2000). Motion blanace filtering. *Eurographics 2000, Computer Graphics Forum*, 19(3):437–446.

Takanishi, A., Egusa, Y., Tochizawa, M., Takeya, T., and Kato, I. (1988). Realization of dynamic walking stabilized with trunk motion. In *CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, pages 68–79.

Ude, A., Man, C., Riley, M., and Atkeson, C. (2000). Automatic generation of kinematic models for the conversion of human motion capture data into humanoid robot motion. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Cambridge, MA.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man. Mach. Syst.*, 10(2):47–53.

Yamane, K. and Hodgins, J. (2009). Simultaneous tracking and balancing of humanoid robots for imitating human motion capture data. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2510–2517, St. Louis.

Yamane, K. and Hodgins, J. (2010). Control-aware mappping of human motion data with stepping for humanoid robots. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 726–733, Taipei, Taiwan.

Yamane, K. and Nakamura, Y. (2003). Dynamics filterconcept and implementation of on-line motion generator for human figures. *IEEE Trans. Robot. Automat.*, 19(3):421–432.

Yamane, K. and Nakamura, Y. (2008a). Dynamics simulation of humanoid robots: forward dynamics, contact, and experiments. In *The 17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*.

Yamane, K. and Nakamura, Y. (2008b). A numerical robust LCP solver for simulating articulated rigid bodies in contact. In *Robotics: Science and Systems*.

Zheng, Y. and Chew, C.-M. (2009). Distance between a point and a convex cone in n-dimensional space: computation and applications. *IEEE Transactions on Robotics*, 25(6):1397–1412.

Zheng, Y., Lin, M. C., and Manocha, D. (2010). A fast n-dimensional ray-shooting algorithm for grasping force optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1300–1305, Anchorage, Alaska.

Zheng, Y., Lin, M. C., and Manocha, D. (2012). On computing reliable optimal grasping forces. *IEEE Transactions on Robotics*, 28(3):619–633.

Zheng, Y. and Qian, W.-H. (2009). Improving grasp quality evaluation. *Robotics and Autonomous Systems*, 57(6-7):665–673.

Zheng, Y. and Yamane, K. (2011). Ball walker: A case study of humanoid robot locomotion in non-stationary environments. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 2021–2028, Shanghai, China.

Zheng, Y. and Yamane, K. (2013a). Human motion tracking control with strict contact force constraints for floating-base humanoid robots. In *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Atlanta, GA.

Zheng, Y. and Yamane, K. (2013b). Ray-shooting algorithms for robotics. *IEEE Transactions on Automation Science and Engineering*, 10(4):862–874.

Zordan, V. and Hodgins, J. (2002). Motion capture-driven simulations that hit and react. In *Proc. ACM SIGGRAPH Symp. Computer Animation*, pages 89–96, San Antonio, TX.