UNIVERSITÀ
DEGLI STUDI DI
GENOVA

PhD Program in Bioengineering and Robotics

Motion Planning and Control of Dynamic Humanoid Locomotion

by

Songyan XIN

Thesis submitted for the degree of *Doctor of Philosophy* (31° cycle)

December 2018

Dr. Nikos G. Tsagarakis — Supervisor

_____ — Head of the PhD program

Thesis Jury:

Barkan Ugurlu, *Özyeğin University* — External examiner

Mingguo Zhao, *Tsinghua University* — External examiner

_____ — Internal examiner

Abstract

Inspired by human, humanoid robots has the potential to become a general-purpose platform that lives along with human. Due to the technological advances in many field, such as actuation, sensing, control and intelligence, it finally enables humanoid robots to possess human comparable capabilities. However, humanoid locomotion is still a challenging research field. The large number of degree of freedom structure makes the system difficult to coordinate online. The presence of various contact constraints and the hybrid nature of locomotion tasks make the planning a harder problem to solve. Template model anchoring approach has been adopted to bridge the gap between simple model behavior and the whole-body motion of humanoid robot. Control policies are first developed for simple template models like Linear Inverted Pendulum Model (LIPM) or Spring Loaded Inverted Pendulum(SLIP), the result controlled behaviors are then been mapped to the whole-body motion of humanoid robot through optimization-based task-space control strategies. Whole-body humanoid control framework has been verified on various contact situations such as unknown uneven terrain, multi-contact scenarios and moving platform and shows its generality and versatility. For walking motion, existing Model Predictive Control approach based

on LIPM has been extended to enable the robot to walk without any reference foot placement anchoring. It is kind of discrete version of "walking without thinking". As a result, the robot could achieve versatile locomotion modes such as automatic foot placement with single reference velocity command, reactive stepping under large external disturbances, guided walking with small constant external pushing forces, robust walking on unknown uneven terrain, reactive stepping in place when blocked by external barrier. As an extension of this proposed framework, also to increase the push recovery capability of the humanoid robot, two new configurations have been proposed to enable the robot to perform cross-step motions. For more dynamic hopping and running motion, SLIP model has been chosen as the template model. Different from traditional model-based analytical approach, a data-driven approach has been proposed to encode the dynamics of the this model. A deep neural network is trained offline with a large amount of simulation data based on the SLIP model to learn its dynamics. The trained network is applied online to generate reference foot placements for the humanoid robot. Simulations have been performed to evaluate the effectiveness of the proposed approach in generating bio-inspired and robust running motions. The method proposed based on 2D SLIP model can be generalized to 3D SLIP model and the extension has been briefly mentioned at the end.

Acknowledgements

First, I would like to thank my advisor Dr. Nikos Tsagarakis for his support and guidance throughout my whole PhD period. He gave great freedom for me to explore what I am interested in. With so much freedom, sometimes people can get lost. At such tough moments, he can always give me some advice to guide me through it. I think that's what a good mentor can do.

There are so many people I want to express my gratitude during the past few years for supporting me. Before starting my PhD at IIT, I did one year internship here, thanks a lot for Dr. Zhibin Li, giving me the chance to start this amazing journey. Joining IIT, I made a lot of friends and learned a lot from them. Especially the members from locomotion group, Dr. Chengxu Zhou, Dr. Przemyslaw Kryczka, Dr. Wesley Roozing, Dr. Juan Castano, Dr. Yangwei You. Among all, I would like to thank Yangwei You for many fruitful discussions and many late hours we spend in the lab. I am also grateful for all members in the lab. Many thanks to Brian Delhaisse for his help on programming and machine learning related topics. Without the help of software and hardware support from the team, many things wouldn't be possible. Thanks to Dr. Enrico Mingo, Luca Muratore, and Arturo Laurenzi for their great

works on software integration. Without post-docs Jorn Malzahn, Navvab Kashiri and technicians Lorenzo Baccelliere, Phil Edward Hudson, Marco Migliorini, Alessio Margan and many others, the robot would never be able to move. Two projects I was involved in during my PhD are WALK-MAN and CogIMon, It is my honor to be part of these big projects.

At the end, I would like to thank my parents for their understanding and unconditional support such that I could spend these years to explore what I am interested in.

Table of contents

Chapter 1

Introduction

1.1 Motivation

Humanoid robotics is directly inspired by human itself. With human-like kinematics, humanoid robots has the potential to become a general-purpose platform that lives along with human. Therefore, the field has received increasingly attention during the past years. The recent DARPA Robotics Challenge (DRC) [1] shows their potential utilities in disaster response scenarios. Other emerging applications include logistic, nursing-care and entertainment. Due to the technological advances in many field, such as actuation, sensing, control and intelligence, it finally enables humanoid robots to possess human comparable capabilities.

However, humanoid robotics is still a challenging research field. Among all kinds of challenges, robust locomotion is perhaps the most basic and challenging one. Almost no humanoid robots could come out of their lab and work with human at this moment.

Many demonstrated motions usually involves large amount of tunning work and are limited in certain scenarios. The robots are not quite ready for the changeable real daily life environment. Most of the problem comes from their high degree-of-freedom structure and relatively small support feet although sometimes the robot benefits from it in terms of mobility and agility. Large number of degree-of-freedom structure in general makes the system difficult to coordinate in real-time. Limited and not continuously available support areas provides great challenges for maintaining balance. Due to friction-limited and unilateral contacts between the feet and the ground, the robot can not accelerate in any direction even if unlimited actuator has been equipped. Moreover, the simple walking is actually a dynamic stable motion which involves multiple phases. Different dynamics are presented in each phase and this complicate the control. For certain period, such as heel-strike and toe-off, the robot is under-actuated and it is actually purposely falling to its next support foot. Above mentioned features add significant difficulties to the dynamic motion planning and control.

Most modern humanoid make use of the zero-moment point (ZMP) criterion to help generating walking motion, such as ASIMO and HRP-2 in Figure 1.1. By controlling the ZMP (or COP) of the robot always inside the convex hull formed by the feet, the humanoid is effectively fully actuated. This largely simplify the planning and control problem. Since this instantaneous stability criterion can be satisfied at any phase of walking motion (single support phase or double support phase), the dynamics can be simplified to a uniform one. While the theory is well established on flat terrain, it can not be naturally generalized to uneven terrain. Assuming fully-actuation throughout

Fig. 1.1 Position controlled humanoid robots. On the left side is ASIMO humanoid [2] developed by Honda. On the right side is HRP-2 humanoid [3], the project is supported by Ministry of Economy, Trade and Industry of Japan.

the motion makes it impossible to be applied to running motion of which the flight phase is one of the most important characteristic and is basically under-actuated.

Industrial robots are probably the most widely used robots in the world and it is actually the most important driver for robotics research. Fast and accurate positioning is the most basic requirement for manufacturing applications so that many industrial robots are designed in such a way to full fill it. They are usually powered by high speed motor followed by high gear ratio transmission system which makes them very rigid. With similar actuation design, early humanoid robots are mostly position controlled and therefore moves in an unnatural way. Pure position control may cause a rise of contact force between robot and environment due to unavoidable modeling error, and leads to unstable behavior. To be truly useful in human's daily environment, humanoid

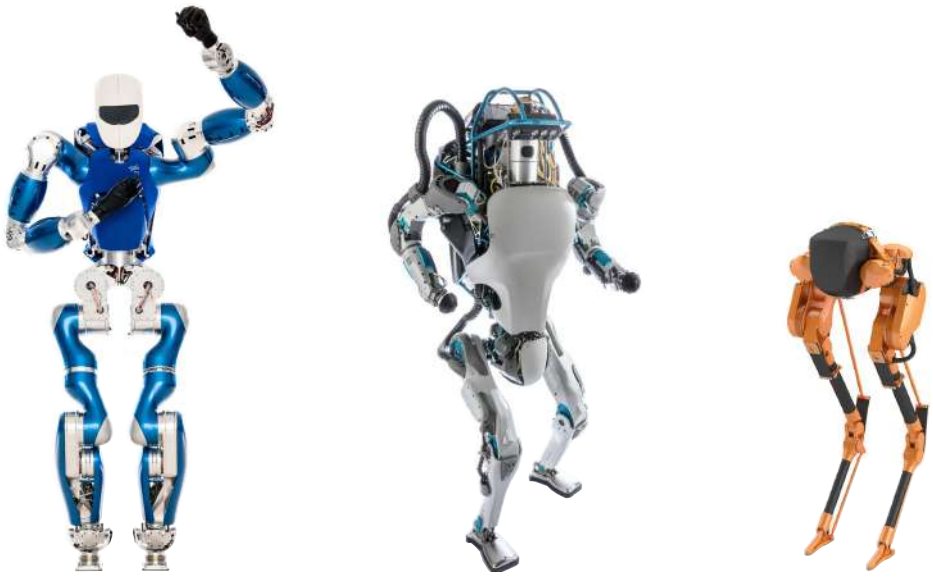


Fig. 1.2 Torque controlled humanoid robots. From left to right, TORO[4] developed by DLR, Atlas [5] developed by Boston Dynamics and CASSIE developed by Agility Robotics.

robots needs to be more compliant, adaptive and energy efficient. With compliance, the robot can interact with human in a safer way. More than that, compliance allows the robot to handle the physical contact between the robot and the environment in a more gentle way which is actually the key for the success of locomotion task. All these requirements demand dynamic control. Last but not least, dynamic locomotion takes advantage of the natural dynamics of the robot and this makes the motion much more energy efficient. Based on above observation, a whole new class of robots which are compliant and dynamic controllable have been build in recent years.

Humanoid robots is intrinsically a high-dimensional, non-linear, dynamical system. It is difficult to directly control all degree of freedom (DoF) of the robot. What's more, many tasks for humanoid robots to achieve are intuitively defined in Cartesian space instead of joint space. This is where template anchoring approach comes into use, it

serves as a bridge which brings the gap between simple model and whole-body model. Simple models that focus on a few meaningful degree of freedoms (DoFs), such as center of mass (CoM), contact points can largely describe the essence part of humanoid dynamics. The idea of template anchoring naturally implies a multi-level control strategy. At higher level, simplified behaviors (such as CoM and Feet movements) are generated based on template model. The lower level controller takes these references and generate whole-body motion based on it. This approach achieves great successes in humanoid locomotion research.

Is there a universal template model for all human behaviors? At least at this moment, the answer is not clear. In the walking research field, the most widely used template model is the Linear Inverted Pendulum Model (LIPM) and it leads to tremendous successes on real humanoid robots such as the two shown in Figure 1.1. The Spring Loaded Inverted Pendulum (SLIP) model has been often used as template model for hopping and running. A novel walking template model Dual Spring Loaded Inverted Pendulum (Dual-SLIP) model has been proposed recently and it tries to unify all types of motion within one framework. Bearing in mind the question asked at the beginning of this paragraph, this dissertation tries to answer it from a different perspective. Assuming there are multiple template models involved in human motion, there needs to be a synthesis-planner to manage the employment of the appropriate template for a certain type of movement as well as the transferring between different models.

Motivated by this, this dissertation presents a unified control framework that can handle a variety of terrain situations as well as different levels of external disturbances. For extremely dynamic motion such as hopping and running, a neural-network based SLIP controller has been introduced.

1.2 Previous Work

The area of dynamic locomotion has been studied extensively over the last decades. Researchers have proposed many different approaches to address the problem. Here not all of them will be reviewed but most popular ones will be covered.

1.2.1 Passive Dynamic Walking

For perusing energy efficiency, passive dynamic walker has been developed. It has no actuation and is able to walk down a slope relying solely on gravity. The demonstrated human-like and efficient walking motion has attracted much attention [6, 7]. Minimal actuation added to the robot makes this type of robot be able to walk on flat terrain [8–11]. More recently, the Cornell Ranger [12] walked 65.2 km in about 31 hours without being interrupted with a single battery charge. The total cost of transport is 0.28 which is similar to the human's one 0.3 and this might eventually leads to energy-efficient walking on more complex robots.

1.2.2 Hybrid-Zero-Dynamics Control

Hybrid-Zero-Dynamics (HZD) [13] is one of the most systematic approach to design walking controller for underactuated robots. In [14], it has been shown that the stability of the swing phase of a three-link planner walking motion is governed by its zero dynamics through partial feedback linearization. The basic idea is to design the zero dynamics for the robot through an indirect manner so that it leads to the achievement of a given task, particularly, without consideration of impact model. By incorporating the impact model into the zero dynamics, the concept of HZD has been introduced [15]. Based on the HZD model, a systematic way to design a feedback controller that leads to stable walking motion has been introduced [16].

1.2.3 Heuristic Control

The best example of this type of controllers is the one from Marc Raibert. Initially, it has been found that a three-part controller could stabilize the hopping motion of a simple hopper consists of a body and a leg [17]. The controller decompose the control into three targets: hopping height, forward speed and posture. Among them, the speed target is regulated by foot placement heuristics which is also the key for balance control. The same idea has been extended to planar bipeds [18] and 3D bipeds [19]. In another work [20], simple heuristic strategies have been applied to simulated humanoid robot to generate omni-walking motion and demonstrates robustness to disturbances from all direction.

1.2.4 Trajectory Optimization

In fact, trajectory optimization is one of the earliest approaches to generating walking and running motions [21]. The idea is to use trajectory optimization to find out the walking motion that satisfying certain objectives and at the same time subject to all kinds of constraints. The problem of this approach in early days is that it is too computationally demanding due to the complexity of robot dynamic model. The optimization based on full humanoid dynamic model is too computationally demanding for early days computer, this approach takes off more recently due to the advances of computer science [22]. Different control parameterizations exist: joint trajectories parameterized as way points [23, 24], piecewise linear function of time [25, 26], B-Splines [27], cubic splines [28, 29]. In general, trajectory optimization approach is time consuming (not ready for real-time application) and tend to have multiple local minimums due to the nonlinearity of the problem.

1.2.5 Simple-Model-Based Control

Instead of directly dealing with the humanoid whole-body model, an alternative is to just focus on a reduced ordered subsystem and the rest part can be more or less independently controlled. The subsystem could be represented by a "template" model [30]. By studying the simplified model dynamics, it gives insights of the whole system.

The most commonly used template for walking motion is Linear Inverted Pendulum Model (LIPM). It has been proposed in [31] and later extended to 3D [32]. The model gives the relationship between CoM and zero moment point (ZMP) [33]. Based on

this model, "ZMP Preview Control" has been proposed to generate humanoid walking motion [34]. Model predictive control (MPC) has been adopted to reformulate the same problem [35] in a more general sense and the resulted controller is more robust due to fast online iterative optimization. In a later work [36], online foot placement adjustment has been incorporated into the MPC control framework. By adding a flywheel on top of the LIPM, this augmented model has been used to study the push recovery property of humanoid robot [37–39].

The Spring Loaded Inverted Pendulum (SLIP) is first introduced in the paper [30] and has long been recognized as a template model for running motion for a wide variety of species [40]. There is no doubt that it also been applied on humanoid robot to generate hopping or running motion [41–46]. In computer graphic field, running motion has been generated by combining LIPM in horizontal direction and spring mass model in vertical direction [47].

1.2.6 Task-Space Control

Task-Space control, also called operational-space control, provides the framework to design the whole-body motion of humanoid robot in task space. Tasks are often intuitively defined in task space rather than joint space. For instance, to grasp an object, the task is usually expressed as the position and orientation of the hand in Cartesian space [48]. Locomotion tasks are often defined by CoM trajectories and feet trajectories [49]. In such a way, the design is carried out in a smaller space with less DoF comparing to corresponding joint space.

Comparing to fixed based manipulator, task space motion generation on humanoid robot (typically floating based robot) has to pay more attention on stability issues. There are many proposed criteria which can be roughly sorted as two categories: instantaneous stability criteria and long-term stability criteria. For static tasks, considering instantaneous stability is usually enough, but this is not the case for dynamic locomotion tasks such as hopping and running. The task-space controller (whole-body controller) only regulates the instantaneous dynamics which implies that without the help of a extended forward looking controller, it is almost impossible for it to achieve dynamic movements alone. As a result, the task-space controller often serves as a low-level controller that takes the task trajectories as input and generate whole-body joint control outputs at current time step.

Virtual Model Control (VMC) [50] is a motion control framework that uses virtual components to create virtual forces and then maps these forces to joint torques through Jacobian transpose. However, the mapping ignores velocity terms (Coriolis and centrifugal terms) of the dynamics of the robot which makes it difficult to be directly applied on very dynamic motions. Considering full-body dynamics, Dynamic Balance Force Control (DBFC) [51] is proposed to calculate full-body joint toques based on desired CoM motion and contact forces without considering the limits on joint torques and accelerations. Seamlessly integrating VMC into DBFC framework also has been presented in the same work. Null-space projection method has been used to control whole-body behaviors without considering of unilateral constraints [52] [53]. To take into account of whole-body dynamics and all sorts of constraints which

are missing from previously mentioned methods, quadratic programming (QP) based optimization method has been proposed [49] [54] [55] [56]. In general, two formulations exists to resolve the multi-objectives QP optimization problem: prioritized approach [49] and weighted approach [55] [56]. The weighted approach tries to resolve the conflicts between multiple tasks by defining a simple scalar weight for each task. The prioritized approach imposes strict priority on each task which means that lower priority task would not interfere higher priority tasks at all in which sense it is similar to the null-space projection method but considering constraints. Actually, the two approaches could be integrated into a unified multi-level framework which considering weights between the same level tasks and hierarchy between different level tasks. Several follow-up works come up with better formulation to reduce the computation cost. A reduced formulation for multiple rigid planar contact scenario has been proposed in [57]. Through decomposition of the equation of motion into floating-base part and direct actuation part [58], the optimization variables could be reduced with a number of n (the number of DoF of the robot) and the resulted optimization could be implemented for real-time control (1 kHz). In [59], a reformulation of prioritized task-space control as conic optimization problem has been proposed and it can be solved at real-time rates. A custom active-set algorithm [60] is developed by exploiting sparsity and temporal structure in the optimization and its performance surpass the best available off-the-shelf solvers. Recently, it has been shown that regulating centroidal angular momentum [61] is very important for dynamic motions [62], as a result, it has emerged to become an important task in the whole-body control framework [24] [45] [63].

1.3 Organization and Contributions

This dissertation aims to the development of dynamic motions for humanoid robots. A framework is provided integrating motion planning, control and potentially learning. Various practical applications based on the framework have been demonstrated to prove its effectiveness. Chapter 2 describes the general humanoid dynamic model and control framework. Several aspects of the dynamic model have been described: joint-space dynamics, floating-base dynamics (system dynamics) and centroidal dynamics. For centroidal dynamics, its relationship with generalized acceleration and ground reaction forces have been described. Since computational dynamics approach has been used, the framework remains general to several existing humanoid robots in the lab. The simulation environment used in this dissertation has also been introduced in the same chapter.

Chapter 3 focus on the optimization based whole-body control. Few important control tasks have been listed. Several applications have been demonstrated to show that how it could be applied to control humanoid whole-body motion. Among the applications, the wall-pushing and recovery example shows potential extension of the whole-body control to multi-contact scenario or quadruped. The self-balancing mobile platform riding case shows the dynamic interaction capability of the controller.

Chapter 4 focus on the walking motion. Improved Model Predictive Control (MPC) formulation has been proposed allowing the robot to walk without any reference foot step anchoring. It can be considered as a discrete version of “walking without thinking”. Based on this proposed planning scheme, versatile walking mode can be achieved.

Depending on how much external force been applied on the robot, the robot could achieve guided walking or reactive stepping. Considering unstructured environment, two cases have been demonstrated: uneven terrain and unknown obstacle. Thanks to the proposed foot step planning, the robot remains stepping in place without falling when its body has been blocked by the obstacle. More challenging walking on line feet and even point feet have been tested with this planner, the results prove that the framework can be extended to under-actuation cases. To the author's best knowledge, the same planning scheme has never been extended to full scale humanoid robots with line feet or point feet. In this last part of this chapter, two new configurations have been proposed to enable the robot to achieve cross-step motion which extensively enlarge the push recovery capability of the humanoid robots.

Chapter 5 extends the motion capability of the robot to hopping and running with new template model. To form the baseline, Marc Raibert three-part controller has been mapped to the whole-body humanoid robot. A data-driven neural-network controller has been proposed to encode the SLIP model behavior and used to online to give reference foot placement. The new proposed controller gives better running velocity regulation comparing to the baseline. Extension to 3D-SLIP model has been described to show the generality and extensibility of the approach.

Chapter 6 concludes and introduces few future extensions emerged from the dissertation.

Chapter 2

Humanoid Dynamic Model and Control Framework

2.1 Humanoid Dynamic Model

This modeling part is based on the book *Rigid Body Dynamics Algorithms* [64] written by Roy Featherstone. Multiple libraries have been developed separately based on the conventions and algorithms described in the book. They are available in different programming languages: Spatial_v2 [65] implemented in Matlab, RBDL [66] in C++ (Python wrapper provided).

To calculate the dynamics of a given rigid-body system, two approaches exists: *symbolic* approach and *computational* approach. Symbolic approach first develop the symbolic equations of motion for the system and then substitute numbers into those equation during dynamic calculation. Computational approach construct a system

model first and supply it as an argument to a model-based dynamics calculation routine (code). In this paper, the second approach is preferred since multiple humanoid robots exists in our lab, it is much easier to construct and modify system model than work out custom symbolic equation of motion for each robot.

2.1.1 Humanoid Robots

Several humanoid platforms have been developed in the lab during the past years. As shown in Figure 2.1, they are COMAN [67], WALK-MAN [68] and CogIMon [69]. These humanoid robots are initially developed with different purpose in mind. COMAN has been developed to explore actuation system with passive compliance. The goal of WALK-MAN is to develop a robotic platform that can operate in unstructured environments and potentially been used for disaster response. The CogIMon project aims at more robust, dependable interaction capabilities between human and robot, in particular the compliant robot CogIMon.

Although developed in different forms, all of them share a similar anthropomorphic structure. Detailed joint configuration of previously mentioned humanoid robots are listed in Table 2.1. From the table it is obvious that all the robots has identical lower-body (Legs) arrangement but with minor difference in their upper-body (Arms and Waist). Since this dissertation focuses on locomotion study of humanoid robots, and also for simplicity, only lower-body is considered for modeling. Without loss of generality, the lower-body which consists of two legs and a common base can be

Fig. 2.1 Humanoid robots developed at Italian Institute of Technology (IIT). From left to right: COMAN, WALK-MAN and CogIMon.

considered as a minimum kinematic tree structure with two branches. Arm and head branch can be easily added without changing the general modeling process.

Since many humanoid robots exists, a unified way to represent those robots is needed. In our lab, the Unified Robot Description Format (URDF) has been used to describe the robot structure. Another reason for using URDF is that it is a self-contained format which includes all robot information and does not depends on any kinematics or dynamics libraries. This is beneficial for external collaborators who want to try their algorithms on our robot model, as well as for internal users who want to try different kinematics or dynamics libraries on the same robot. To summarize, the model representation is decoupled from later usage. For a specific user who is using a specific software to model and simulate these robots, only a parser is needed to translate URDF model to their own model structure and this is usually provided by most off-the-shelf libraries.

Table 2.1 Humanoid Robots Joint DoFs Comparison

		Humanoid Robots		
		COMAN	WALK-MAN	CogIMon
Arm	Shoulder	3	3	3
	Elbow	1	1	1
	Wrist	N/A	3	3
Waist		3	2	2
Leg	Hip	3	3	3
	Knee	1	1	1
	Ankle	2	2	2

Note: Neck and Hand joints are not listed here.

2.1.2 Kinematic Tree

A humanoid robot is basically a floating-base multi-rigid-body system and its bodies are connected resembling a kinematic tree-structure. A data structure defining a kinematic tree should contains at least the following information:

- the number of bodies (the number of joints is the same)
- connectivity data (describing how the bodies are connected together)
- joint data (includes joint type and joint parameters)
- geometry data (the location of each joint in each body)
- body data (the spatial inertia of each body)

Here, we will mainly describe how the connectivity graph is defined given the lower-body of a humanoid robot.

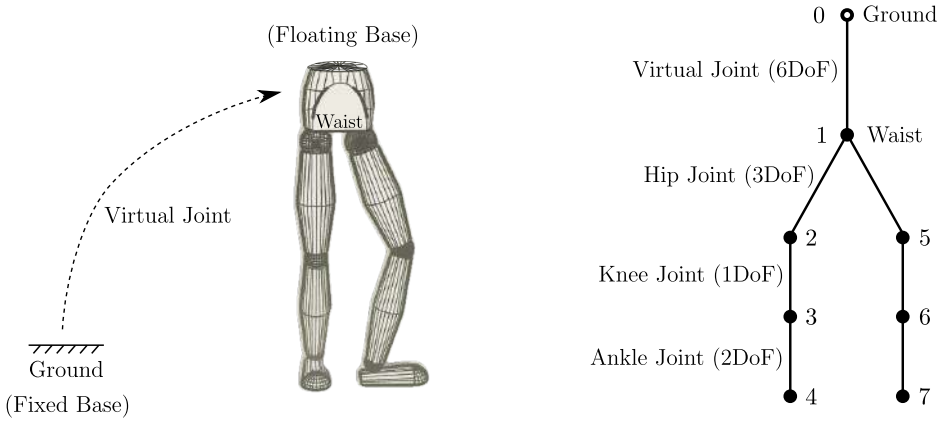


Fig. 2.2 Connectivity graph of the lower-body of humanoid robot. Nodes represent bodies, in particular, the root node is labeled with a special empty circle. Arcs represent joints and they connect all bodies together. Body indexes have been labeled next to each node.

Connectivity Graph

A connectivity graph describes the component parts (nodes and arcs) and how they are connected together. In this case, the nodes represents bodies and arcs represent joints. Exactly one node is defined as the base and the rests represent moving bodies. For a humanoid robot, waist body is usually taken as the floating-base and connected to the fixed base (ground) through a 6-DoF virtual joint as shown in Figure 2.2. The 6 DoF virtual joint does not impose any motion constraints on the robot and it is defined to identify the position and velocity of the floating base with respect to ground.

Numbering Scheme

The numbering of bodies is defined here. The fixed base is numbered as Body 0 (ground in this case). The floating base is selected as Body 1 (waist of the robot). The rest bodies are numbered from 2 to N_B (number of bodies) in a order such that each body

has a higher number than its parent body and this is called forward numbering. Arcs are numbered such that arc i connects node i to its parent node, as a result, arc index starts from 1.

In our case, the numbering starts from left upper leg to left lower leg and continue with right upper leg, then right lower leg. Actually, due to the absence of spherical joint type and universal joint type in URDF definition, the 3 DoF hip joint and 2 DoF ankle joint have been decomposed into multiple sequential single DoF revolute joints which are connected through intermediate bodies. Take the robot CogIMon for example, the connectivity graph is plotted in Figure 2.3. All joints (arcs) have been labeled out in the graph. "Ground" and "Waist" body have been labeled out since they represent two important bases of the graph. All the other bodies (nodes) are named by combining its parent joint name and "Body" suffix. For example, the body after "LHipRoll" joint is named as "LHipRollBody".

Representation of Connectivity Graph

There are many ways to represent a connectivity graph. One way is to keep joint predecessor array p and successor array s in a pair, such that $p(i)$ and $s(i)$ are predecessor body index and successor body index of joint i. Another way it the parent array λ. Its element $\lambda(i)$ represents the parent body number of body i. For example, the connectivity graph in Figure 2.3, the parent body of body 1 is body 0: $\lambda(1) = 0$, the parent body of body 2 is body 1: $\lambda(2) = 1$. Especially, the parent body of body 8 is also body 1: $\lambda(8) = 1$ since the tree starts branching from body 1. Similarly,

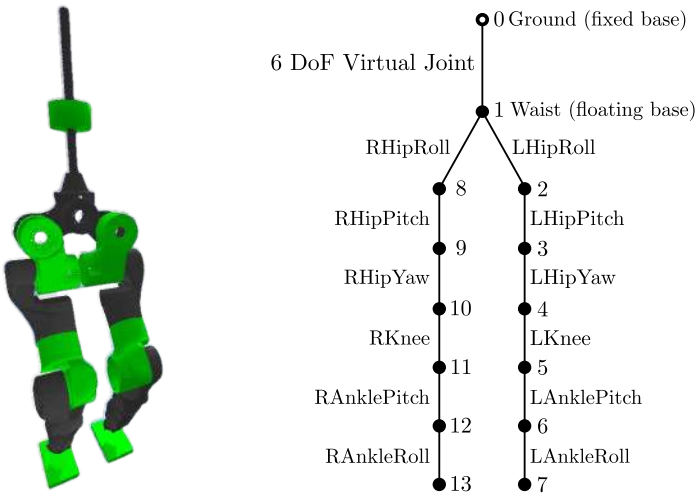


Fig. 2.3 Connectivity graph of the lower-body of CogIMon robot.

we can figure out all elements and organize them together to form the parent array: $\lambda = [0, 1, 2, 3, 4, 5, 6, 1, 8, 9, 10, 11, 12]$ with $1 \leq i \leq N_B$. The parent array provides complete description of the connectivity graph and most algorithms only depends on it.

2.1.3 Joint Space Dynamic Model

Here the equation of motion is defined in joint space. We will start with the configuration parameterization of the robot. $\boldsymbol{q}$ is defined as a full description of the configuration of the system. The floating base of the robot is connected to the ground through a 6 DoF joint as introduced in the previous section. The 6 DoF free flying joint $\in SE(3)$ normally consists of a 3 DoF translational joint and a 3 DoF spherical joint. Three scalar parameters $\boldsymbol{p}_f = [p_f^x, p_f^y, p_f^z] \in \mathbb{R}^3$ is enough to define the translational

joint configuration. For the rotational part $\in SO(3)$, there are several different parameterizations exist: rotation matrix, Euler angles, quaternions. To avoid joint singularities and also keep minimum parameterization, quaternions $\boldsymbol{\epsilon}_f = [\epsilon_f^w, \epsilon_f^x, \epsilon_f^y, \epsilon_f^z]$ has been chosen to defined the orientation of the floating base. In such a way, a total of 7 parameters is needed to represent the pose of the floating base with respect to the ground. Assuming a robot has n actuated joints and their configuration represented by $\boldsymbol{q}_a \in \mathbb{R}^n$, the whole configuration $\boldsymbol{q}$ is defined in the space $\mathbb{R}^{n+7}$ by combining $\boldsymbol{p}_f$, $\boldsymbol{\epsilon}_f$ and $\boldsymbol{q}_a$.

The generalized velocity of the system is defined as $\dot{\boldsymbol{q}} \in \mathbb{R}^{n+6}$, it consist of two part: the generalized velocity of the floating base $\dot{\boldsymbol{q}}_f \in \mathbb{R}^6$ and the rates of all actuated joints $\dot{\boldsymbol{q}}_a \in \mathbb{R}^n$. The system acceleration is denote as $\ddot{\boldsymbol{q}} \in \mathbb{R}^{n+6}$ and share similar form as velocity. The joint space dynamic equations of motion takes the standard from:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{S}_a\boldsymbol{\tau} + \boldsymbol{J}_s(\boldsymbol{q})^T\boldsymbol{F}_s, \tag{2.1}$$

where $\boldsymbol{H} \in \mathbb{R}^{(n+6)\times(n+6)}$, $\boldsymbol{C}\dot{\boldsymbol{q}} \in \mathbb{R}^{n+6}$ and $\boldsymbol{G} \in \mathbb{R}^{n+6}$ are the mass matrix, velocity term (Coriolis and centrifugal forces) and gravitational term. The matrix $\boldsymbol{S}_a = [\boldsymbol{0}_{n\times 6}, \boldsymbol{1}_{n\times n}]^T \in \mathbb{R}^{(n+6)\times n}$ is a selection matrix for the actuated joints. $\boldsymbol{\tau} \in \mathbb{R}^n$ stands for the actuated joint torques. $\boldsymbol{J}_s \in \mathbb{R}^{(6N_s)\times(n+6)}$ is a concatenated support Jacobian and it depends on the number of supports N_s, $\boldsymbol{F}_s \in \mathbb{R}^{6N_s}$ is the concatenated ground

reaction forces (GRF):

$$\boldsymbol{J}_s = \begin{bmatrix} \boldsymbol{J}_s^1 \\ \boldsymbol{J}_s^2 \\ \cdots \\ \boldsymbol{J}_s^{N_s} \end{bmatrix}, \quad \boldsymbol{F}_s = \begin{bmatrix} \boldsymbol{f}_s^1 \\ \boldsymbol{f}_s^2 \\ \cdots \\ \boldsymbol{f}_s^{N_s} \end{bmatrix} \tag{2.2}$$

where $\boldsymbol{J}_s^i$ is the Jacobian matrix associated with the support i and $\boldsymbol{f}_s^i$ is the contact force acting on this support.

2.1.4 System Newton and Euler Equations

In [70], it shows that the Newton and Euler equations for the whole system are embedded in the joint space equation of motion of a floating-base articulated rigid body systems. As pointed out in [58], the joint space equation of motion (2.1) can be decoupled into two parts:

$$\begin{bmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_a \end{bmatrix} \ddot{\boldsymbol{q}} + \begin{bmatrix} \boldsymbol{C}_1 \\ \boldsymbol{C}_a \end{bmatrix} \dot{\boldsymbol{q}} + \begin{bmatrix} \boldsymbol{G}_1 \\ \boldsymbol{G}_a \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_{s,1}^T \\ \boldsymbol{J}_{s,a}^T \end{bmatrix} \boldsymbol{F}_s \tag{2.3}$$

where the top six rows with subscript $[\cdot]_1$ represents the floating base part, and the rest part with subscript $[\cdot]_a$ corresponds to the actuated part. Rewriting this equation into two:

$$\boldsymbol{H}_1 \ddot{\boldsymbol{q}} + \boldsymbol{C}_1 \dot{\boldsymbol{q}} + \boldsymbol{G}_1 = \boldsymbol{J}_{s,1}^T \boldsymbol{F}_s \tag{2.4}$$

$$\boldsymbol{H}_a \ddot{\boldsymbol{q}} + \boldsymbol{C}_a \dot{\boldsymbol{q}} + \boldsymbol{G}_a = \boldsymbol{\tau} + \boldsymbol{J}_{s,a}^T \boldsymbol{F}_s \tag{2.5}$$

Eq. (2.4) is the system Newton and Euler equation, it express the change of momentum of the system as a function of external force. This is exactly the centroidal dynamic of the system expressed in the inertial frame. As we can see in the section 2.1.5, with one more coordinate transformation, the centroidal dynamics of the system can be constructed.

2.1.5 Centroidal Dynamic Model

Relationship with Generalized Acceleration

Introduced in [71] [61], Centroidal momentum is the sum of all body spatial momenta computed with respect to the CoM. It collects the system linear momentum and angular momentum together. A Centroidal Momentum Matrix (CMM) $\boldsymbol{A}_G \in \mathbb{R}^{6\times(n+6)}$ is defined and it relates the system generalized velocity $\dot{\boldsymbol{q}}$ to the centroidal momentum $\boldsymbol{h}_G$:

$$\boldsymbol{h}_G = \boldsymbol{A}_G \dot{\boldsymbol{q}} \tag{2.6}$$

The derivative of (2.6) is often required when performing dynamic whole-body control:

$$\dot{\boldsymbol{h}}_G = \boldsymbol{A}_G \ddot{\boldsymbol{q}} + \dot{\boldsymbol{A}}_G \dot{\boldsymbol{q}} \tag{2.7}$$

In [72] [73], it has been shown that the CMM $\boldsymbol{A}_G$ is related to the joint space mass matrix $\boldsymbol{H}$ and can be efficiently calculated:

$$\boldsymbol{A}_G = {}^1\boldsymbol{X}_G^T \boldsymbol{S}_1 \boldsymbol{H} = {}^1\boldsymbol{X}_G^T \boldsymbol{H}_1 \tag{2.8}$$

where $\boldsymbol{H}_1$ is the floating base inertia matrix appears in eq. (2.4), with a floating base selector $\boldsymbol{S}_1 = [\mathbf{1}_{6\times 6}, \mathbf{0}_{6\times n}]$, it could be selected from the joint space inertia matrix $\boldsymbol{H}$. ${}^1\boldsymbol{X}_G^T \in \mathbb{R}^{6\times 6}$ is a spatial transformation matrix from that transfer spatial momentum from the floating base (Body 1) to the CoM:

$${}^1\boldsymbol{X}_G^T = \begin{bmatrix} {}^G\boldsymbol{R}_1 & {}^G\boldsymbol{R}_1 \boldsymbol{S}({}^1\boldsymbol{p}_G)^T \\ \mathbf{0} & {}^G\boldsymbol{R}_1 \end{bmatrix} \tag{2.9}$$

where $\boldsymbol{S}(\boldsymbol{p})$ provides the skew symmetric cross product matrix such that $\boldsymbol{S}(\boldsymbol{p})\boldsymbol{v} = \boldsymbol{p} \times \boldsymbol{v}$. The orientation of Frame G could be selected to be parallel to the ground inertial frame, than the rotation matrix ${}^G\boldsymbol{R}_1 = {}^0\boldsymbol{R}_1$. ${}^1\boldsymbol{p}_G$ stands for the vector from floating base origin to CoM expressed in floating base frame and it can be calculated ${}^1\boldsymbol{p}_G = {}^1\boldsymbol{R}_0({}^0\boldsymbol{p}_G - {}^0\boldsymbol{p}_1)$. The centroidal dynamics velocity-dependent bias $\dot{\boldsymbol{A}}_G\dot{\boldsymbol{q}}$ can be calculated form joint-space velocity term $\boldsymbol{C}\dot{\boldsymbol{q}}$:

$$\dot{\boldsymbol{A}}_G\dot{\boldsymbol{q}} = {}^1\boldsymbol{X}_G^T \boldsymbol{S}_1 \boldsymbol{C}\dot{\boldsymbol{q}} = {}^1\boldsymbol{X}_G^T \boldsymbol{C}_1\dot{\boldsymbol{q}} \tag{2.10}$$

where $\boldsymbol{C}_1\dot{\boldsymbol{q}}$ is the floating base velocity term in eq. (2.4), it also can be selected out from the joint space equation of motion with the selector $\boldsymbol{S}_1$.

Note that, here it is assumed that the floating base spatial velocity ${}^1\mathbf{v}_1 = [{}^1\boldsymbol{\omega}_1^T, {}^1\boldsymbol{v}_1^T]^T$ expressed in its body frame has been chosen as its generalized velocity $\dot{\boldsymbol{q}}_f$.

Relationship with Ground Reaction Force

On one hand, the centroidal dynamics $\dot{\boldsymbol{h}}_G$ is linear related to the generalized acceleration $\ddot{\boldsymbol{q}}$ as shown in (2.7) On the other hand, it is also linked to external forces on the system as stated by Newton's and Euler's laws that the changing rate of momentum $\dot{\boldsymbol{h}}_G$ equals to the net external wrench $\boldsymbol{f}_G$ on the robot:

$$\dot{\boldsymbol{h}}_G = \boldsymbol{f}_G^{net} \tag{2.11}$$

$\boldsymbol{f}_G^{net}$ is comprised of the gravity force and ground reaction force $\boldsymbol{F}_s$ expressed at G. The spatial transform ${}^s\boldsymbol{X}_G^T$ can be used to transform the external wrench from support Frame to G Frame:

$$\dot{\boldsymbol{h}}_G = \boldsymbol{f}_G^{net} = {}^s\boldsymbol{X}_G^T\boldsymbol{F}_s + \boldsymbol{f}_G \tag{2.12}$$

where $\boldsymbol{f}_G \in \mathbb{R}^6$ is the gravity force expressed at G, ${}^s\boldsymbol{X}_G^T \in \mathbb{R}^{6\times 6N_s}$ takes the form:

$${}^s\boldsymbol{X}_G^T = \begin{bmatrix} {}^1\boldsymbol{X}_G^T & {}^2\boldsymbol{X}_G^T & \dots & {}^{N_s}\boldsymbol{X}_G^T \end{bmatrix} \tag{2.13}$$

where each element is a spatial force transformation matrix that transform the each GRF to Frame G:

$${}^i\boldsymbol{X}_G^T = \begin{bmatrix} {}^G\boldsymbol{R}_i & {}^G\boldsymbol{R}_i\boldsymbol{S}({}^i\boldsymbol{p}_G)^T \\ \boldsymbol{0} & {}^G\boldsymbol{R}_i \end{bmatrix} \tag{2.14}$$

2.2 Control Framework

2.2.1 Robot-Environment Loop

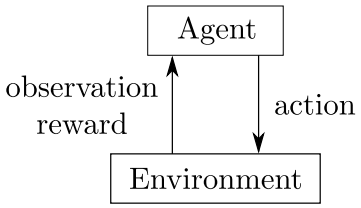


Fig. 2.4 Agent-Environment Loop.

In this part, the overview of the whole control framework is provided. In learning field, reinforcement learning [74] is a computational approach where an *agent* interacts with an *environment* by taking *actions* in which it tries to maximize an accumulated reward. As shown in Figure 2.4, the agent-environment loop summarizes what is going on during the learning process: an agent in a current state takes an action, the environment reacts and responds, returning a new observation and reward to the agent. Given the updated observation and reward, the agent chooses the next action, and the loop repeats until an environment is solved or terminated. An typical implementation of the classic "agent-environment loop" developed by OpenAI Gym [75] is given in Listing 2.1.

Listing 2.1 Agent-Environment Loop Python Example

```
import gym
env = gym.make("Humanoid")
observation = env.reset()
for t in range(1000):
```

```
action = env.action_space.sample()
observation, reward, done, info = env.step(action)
```

In control engineering, the terms *agent*, *environment* and *action* can be replaced by *controller*, *controlled plant* and *control signal*. The same applies to robotics field (Figure 2.5), the controller of a humanoid robot can be considered as the agent. The control plant contains the robot and its surroundings , it could happens in real life or in simulation. Here in this paper, we mainly refer the environment as simulation. The controller receives human commands (such as stand still, walking forward) and generates control commands for each actuator, such as position or torque. After receiving these commands, the robot takes actions immediately. As a result, the robot will interact with the world at contact points. If this happens in a simulator, simulated contact forces will be calculated based on some contact model and exert on the robot as external forces. Then the simulator will step forward one step by integrating the dynamics of the robot. At the end of one step simulation, the states of the robot will be sent back to the controller. Based on updated states, the controller will calculate new control commands accordingly, and the loop goes on and on, until tasks have been accomplished.

2.2.2 Simulation Environment

The full 3D simulation environment that we use in this paper is the mainly Gazebo [76] and PyBullet [77].

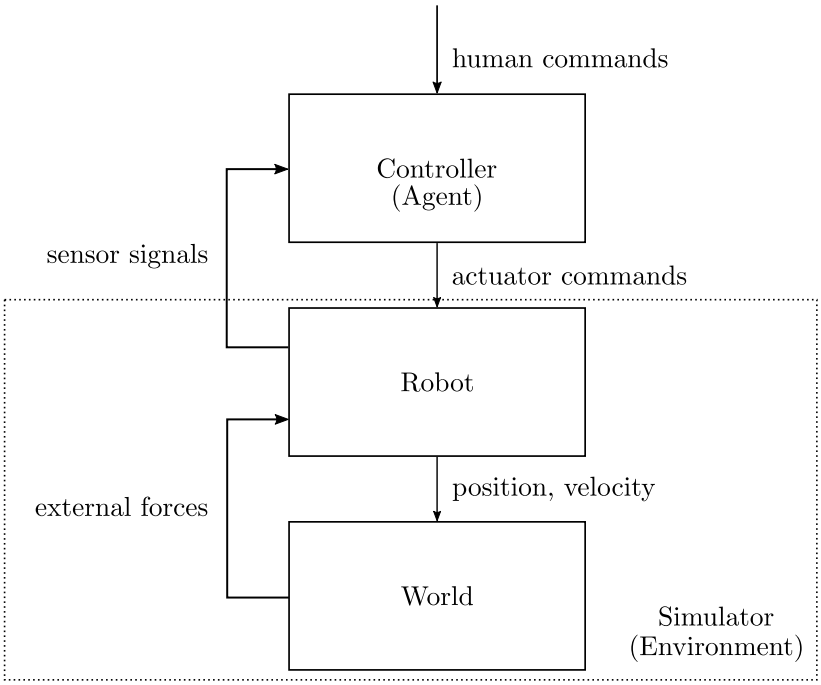


Fig. 2.5 Robot-Environment Loop.

Gazebo is an open source rigid-body simulator that supports multiple choice of physics engines, such as Open Dynamics Engine (ODE) [78], Bullet Physics Engine [79]. It is originally part of the Robot Operating System (ROS) [80] project and later been separated as an independent package. ROS integrates closely with Gazebo through the gazebo_ros_pkgs [81]. This package provides a Gazebo plugin module that allows bidirectional communication between Gazebo and ROS. Simulated sensor and physics data can stream from Gazebo to ROS, and actuator commands can stream from ROS back to Gazebo. The main advantage of using Gazebo/ROS environment is that the controller can be easily switched from simulation to real-world environment since ROS is served as an unified intermediate communication interface between controller and environment. However the main issue of ROS is that it is not a hard real time operating system. This might not be a problem for tasks like manipulation but it is a basic

requirement for tasks like locomotion for which timing matters a lot for balancing capability. In simulation, it is not a big issue since the simulator can be slowed done by modifying its real-time-factor setting. For real world implementation, ROS2 with real-time support has been developed in promising of addressing the issue. In our case, we mainly test our controller in simulation so we don't need to worry much about the timing issue.

Another simulation environment we have tested is PyBullet which is a Python module that extends the Bullet Physics Engine [79] with robotics and machine learning capabilities. Bullet solves the equations of motion for articulated rigid bodies in generalized coordinates while simultaneously satisfying physical constraints including contact, joint limits and actuator models. Then the state of the system is numerically integrated over time using a semi-implicit scheme. The advantage of PyBullet is that you have full control of the simulation process. Meaning that you could step through the simulation by your self, in this way, the timing issue is no more a problem. For the controller, it could take as much as time it needs to calculate the control outputs during which period the simulation is paused. Given the controller outputs, the simulation could be step forward for required time step. The step simulation will perform all the actions in a single forward dynamics simulation step such as collision detection, constraint solving and integration and at the end give back the system states which are needed for next step control outputs calculation.

For experiment, XBotCore [82], a light-weight, Real-Time (RT) software platform has been developed in our lab to enable RT control on real world hardware.

2.2.3 Hierarchical Controller

The main part of this thesis is focusing on the designing of controller for the humanoid robot. A detailed robot environment loop diagram has been shown in Figure 2.6 including the proposed controller structure. The control system has been designed as a hierarchical system consists of multiple levels:

- The highest level is the behavior level, it receives the abstract human commands and decides which template model fits the best for the commanded movement. For example, the LIPM can be used for walking motion but it is not possible to generate hopping and running motion. For the later cases, SLIP model fits better. This level of controller then is responsible for choosing correct template model and switching from current behavior to new behavior. It also plays a important role in push recovery situation which will be discussed in more detail in Chapter 4.

- The middle level is Model Predictive Controller (MPC) based on decided template model. The long-term prediction based on simple model is crucial for robust locomotion due to their long-term stability nature. By optimizing a finite time-horizon evolution of the simple model, the optimal plan is found and the first timeslot of the plan will be fed to the whole-body controller to execute. This optimization process is repeated at every timestep so the robot is reactive to external disturbances through out the whole stage of movement.

- The low level controller is a whole-body dynamic controller that tries to full fill the instantaneous plan given by upper level MPC controller, and at the same time, with respect to all sorts of constraints, such as dynamic feasibility, friction cone, torque limits.

Each level of the control system runs at different frequency. The simulation runs at 1 kHz frequency which means that the simulator will integrate the system dynamics every 1 ms given the joint torque command from the controller. This suggests that it is better to run the whole-body controller at 1 kHz frequency to be able to update torque command together with the simulation loop. Moving upward through the hierarchy, the MPC controller does not necessarily runs at the same frequency as the whole-body controller. It can run at 500 Hz or even at lower frequency. Normally, faster MPC control frequency means shorter update duration which makes the robot more reactive to external disturbances. The highest behavior level controller will only be activated when human command been modified or large external disturbance happens. For example, the human command switch from walking to running, the behavior level controller will switch template model from LIPM to SLIP. Another case is when large push occurs, the robot needs to switch from standing to walking or from walking to hopping or running.

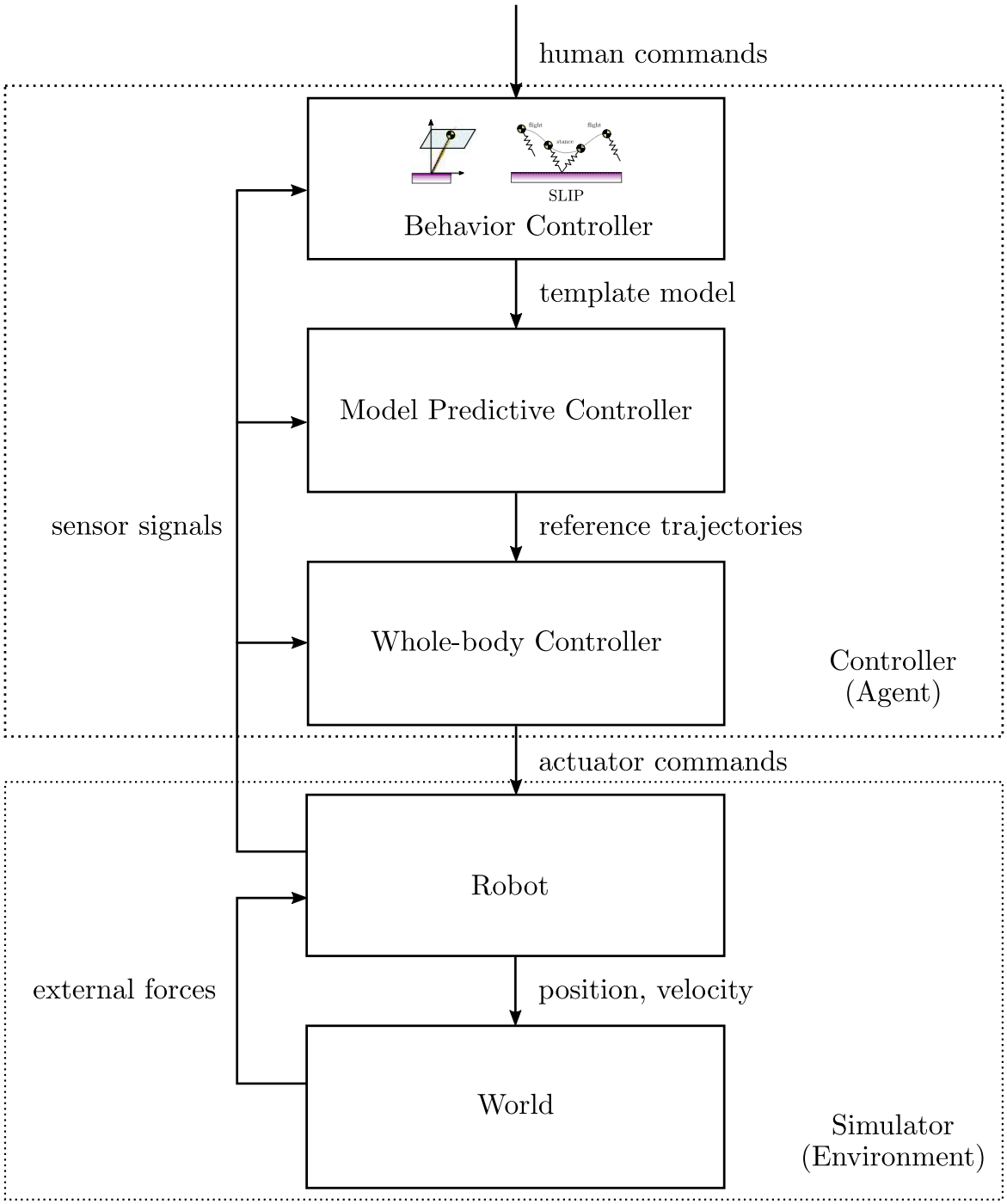


Fig. 2.6 Controller Diagram.

Chapter 3

Whole-Body Controller

3.1 Task-Space Control Problem

The standard whole-body joint space dynamics equation of motion for a humanoid robot is given in 2.1.3, here we repeat it here for reference:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{S}_a^T\boldsymbol{\tau} + \boldsymbol{J}_s(\boldsymbol{q})^T\boldsymbol{F}_s, \tag{3.1}$$

where $\boldsymbol{q}$ is the configuration of the system, $\dot{\boldsymbol{q}} \in \mathbb{R}^{n+6}$ and $\ddot{\boldsymbol{q}} \in \mathbb{R}^{n+6}$ are the generalized velocity and acceleration of the system. $\boldsymbol{H} \in \mathbb{R}^{(n+6)\times(n+6)}$, $\boldsymbol{C}\dot{\boldsymbol{q}} \in \mathbb{R}^{n+6}$ and $\boldsymbol{G} \in \mathbb{R}^{n+6}$ are the mass matrix, velocity-terms (Coriolis and centrifugal forces) and gravitational term. The matrix $\boldsymbol{S}_a = [\boldsymbol{0}_{n\times 6}, \boldsymbol{1}_{n\times n}]$ is a selection matrix for the actuated joints. $\boldsymbol{\tau} \in \mathbb{R}^{n+6}$ stands for the actuated joint torques. $\boldsymbol{J}_s \in \mathbb{R}^{(6N_s)\times(n+6)}$ is a concatenated support Jacobian and it depends on the number of supports N_s. $\boldsymbol{F}_s \in \mathbb{R}^{6N_s}$ is the concatenated ground reaction forces (GRFs).

When designing the whole-body behavior of a humanoid robot, it is often more intuitive and convenient to plan the motion in task space (or operational space) than joint space. In general, a task space objective can be defined as:

$$\boldsymbol{r}_t = \boldsymbol{A}_t(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.2}$$

where $\boldsymbol{r}_t$ represents a general task, $\boldsymbol{A}_t$ is the generalized task Jacobian. Here, the task is defined in kinematic level, to incorporated this task into dynamic level control, the second order relationship could be find out by differentiation:

$$\dot{\boldsymbol{r}}_t = \boldsymbol{A}_t\ddot{\boldsymbol{q}} + \dot{\boldsymbol{A}}_t\dot{\boldsymbol{q}} \tag{3.3}$$

Given commanded instantaneous task dynamics, $\dot{\boldsymbol{r}}_{t,c}$, the task space control problem is to find the best possible joint acceleration $\ddot{\boldsymbol{q}}$ to match it while satisfying all constraints.

3.1.1 Ground Reaction Force Constraints

The whole-body task space control often needs to handle multiple tasks who might be conflicts with each other, and at the same time, meets all constraints. Therefore, the problem falls onto two points: how to handle multiple tasks and how to incorporate constraints. Without considering GRFs constraints, prioritized operational space control framework for humanoids in support has been proposed in [83]: original equation of motion has been projected onto a subset of configuration space which is consistent with

the support constraints. In the framework, multiple tasks privatization are handled through a series of nested null-space projectors.

However, the constraints on GRFs $\boldsymbol{F}_s$ (friction limits and unilateral constraint) are crucial since many tasks including centroidal momentum are regulated by GRFs. Violation of these constraints could lead to the failure of all other tasks. Additional treatments are required [70] [84]. Some other works [85] [56] [57] have proposed to use quadratic programming (QP) to solve the multiple-objective task space control problem while with fully respect of GRFs constraints.

To constrain the GRFs and the ZMP (CoP) associated with it, different approaches has been proposed. In [86], it is proposed that each spatial contact force $\boldsymbol{f}_s^i \in \mathbb{R}^6$ could be represented by a combination of pure forces (no moment) that act on the vertices of the convex support region. Another way to handle the spatial contact force is to approximate the friction cone with a friction pyramid as shown in Figure 3.1. Given the friction coefficient μ_i for support i, the corresponding friction cone $\mathcal{C}_s^i$ is defined:

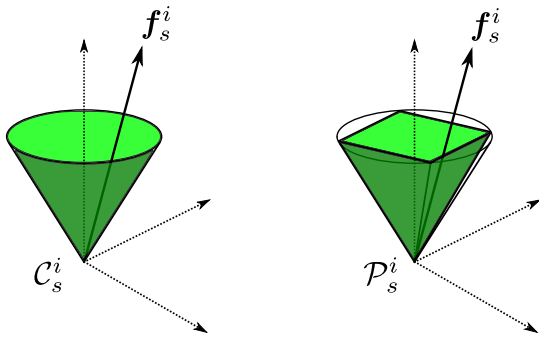


Fig. 3.1 Friction Model. The spatial force $\boldsymbol{f}_s^i$ for support i is bounded by the friction cone $\mathcal{C}_s^i$ and it can be approximated by a pyramid $\mathcal{P}_s^i$.

$$\mathcal{C}_s^i = \{(f_x^i, f_y^i, f_z^i) \in \mathbb{R}^3 \mid \sqrt{(f_x^i)^2 + (f_y^i)^2} \leq \mu_i f_z^i\} \tag{3.4}$$

where (f_x^i, f_y^i, f_z^i) is the force part of the contact spatial force $\boldsymbol{f}_s^i = [n_x^i, n_y^i, n_z^i, f_x^i, f_y^i, f_z^i]^T$. The friction pyramid can be defined as:

$$\mathcal{P}_s^i = \{(f_x^i, f_y^i, f_z^i) \in \mathbb{R}^3 \mid f_x^i \leq \mu_i f_z^i,\ f_y^i \leq \mu_i f_z^i\} \tag{3.5}$$

The friction pyramid is used as friction constraints in this paper since the conic one can not fit into the quadratic optimization. The unilateral constraints can be also defined for the same support:

$$0 \leq f_z^i \tag{3.6}$$

Assuming the feet has a box collision, the ZMP constraints can be also defined:

$$\begin{aligned} d_x^- \leq\ \ & n_y^i / f_z^i \leq d_x^+ \\ d_y^- \leq\ & -n_x^i / f_z^i \leq d_y^+ \end{aligned} \tag{3.7}$$

where the size of the sole is defined by (d_x^-, d_x^+) in x direction and (d_y^-, d_y^+) in y direction, (n_x^i, n_y^i, n_z^i) is the couple part of the contact spatial force $\boldsymbol{f}_s^i = [n_x^i, n_y^i, n_z^i, f_x^i, f_y^i, f_z^i]^T$.

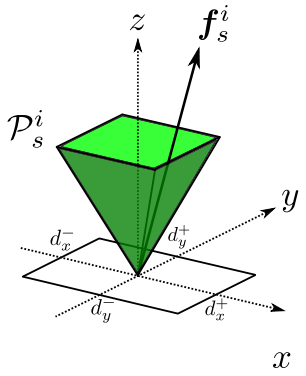


Fig. 3.2 ZMP constraints.

Note that all these constraints related to ground reaction forces are defined in local frame, normally the contact spatial force is defined at the contact point with world orientation and it needs to be rotated into local frame first:

$$ {}^{i}\boldsymbol{f}_s^i = \begin{bmatrix} {}^{i}\boldsymbol{R}_0 & \mathbf{0} \\ \mathbf{0} & {}^{i}\boldsymbol{R}_0 \end{bmatrix} \boldsymbol{f}_s^i \tag{3.8} $$

since the spatial force has just been rotated locally, there is no positional translation involved.

3.1.2 Task-Space Control Formulation

Many works [85] [56] [57] have proposed to use quadratic programming (QP) to solve the multiple-objective task space control problem: joint accelerations $\ddot{\boldsymbol{q}}$ and joint torques $\boldsymbol{\tau}$ and GRFs $\boldsymbol{F}_s$ are considered as optimization variables and all constraints related to them could be integrated into the optimization problem. When there are multiple tasks to be considered, one could either combine all the tasks errors in a weighted-sum form within the cost function, with more important tasks given higher weights, or one could solve the QP for each individual task in a prioritized manner. More specifically, to optimize for lower-priority tasks, the QP may be modified with additional constraints to ensure that all the higher priority tasks are not affected. As a result, a series of nested QPs (or stack of QPs) needs to be solved to find the outputs. The former approach been called "weighted approach" and the latter one is "prioritized approach". Actually, the two approaches could be integrated into a unified

multi-level framework which considering weights between the same level tasks and priorities between different level tasks. Several follow-up works come up with different formulation to reduce the computation cost. Through decomposition of the equation of motion into floating-base part and direct actuation part [58], the optimization variables could be reduced with a number of n (the number of DoF of the robot) and the resulted optimization could be implemented for real-time control (1 kHz). In [59], a reformulation of prioritized task-space control as conic optimization problem has been proposed and it can be solved at real-time rates. In this thesis we use the weighted QP to solve the task space control problem.

As shown in section 2.1.4, the joint space dynamic equation can be partitioned into two parts:

$$\boldsymbol{H}_1\ddot{\boldsymbol{q}} + \boldsymbol{C}_1\dot{\boldsymbol{q}} + \boldsymbol{G}_1 = \boldsymbol{J}_{s,1}^T\boldsymbol{F}_s \tag{3.9}$$

$$\boldsymbol{H}_a\ddot{\boldsymbol{q}} + \boldsymbol{C}_a\dot{\boldsymbol{q}} + \boldsymbol{G}_a = \boldsymbol{\tau} + \boldsymbol{J}_{s,a}^T\boldsymbol{F}_s \tag{3.10}$$

the previous one represents the system Newton and Euler equation (floating base part), the later one stands for the actuated part. From (3.10), it is easy to figured out that $\boldsymbol{\tau}$ is linearly related to $\ddot{\boldsymbol{q}}$ and $\boldsymbol{F}_s$:

$$\boldsymbol{\tau} = \boldsymbol{H}_a\ddot{\boldsymbol{q}} + \boldsymbol{C}_a\dot{\boldsymbol{q}} + \boldsymbol{G}_a - \boldsymbol{J}_{s,a}^T\boldsymbol{F}_s \tag{3.11}$$

It indicates that the joint torque $\boldsymbol{\tau}$ does not need to be included into the optimization variable, it could calculated after finding out proper $\ddot{\boldsymbol{q}}$ and $\boldsymbol{F}_s$, as long as they satisfy the

system dynamic constraints introduced by (3.9). So, we could choose the optimization variable to be $\boldsymbol{X} = [\ddot{\boldsymbol{q}}^T, \boldsymbol{F}_s^T]^T$, the general quadratic programing problem takes the form:

$$\begin{aligned} \min_{\boldsymbol{X}} \ & \frac{1}{2}\boldsymbol{X}^T\boldsymbol{Q}\boldsymbol{X} + \boldsymbol{p}^T\boldsymbol{X} \\ s.t. \ & \boldsymbol{C}_{ineq}\boldsymbol{X} \leq \boldsymbol{c}_{ineq} \\ & \boldsymbol{C}_{eq}\boldsymbol{X} = \boldsymbol{c}_{eq} \end{aligned} \tag{3.12}$$

where costs are defined by $\boldsymbol{Q}$ and $\boldsymbol{p}$, inequality constraints are defined by $\boldsymbol{C}_{ineq}$ and $\boldsymbol{c}_{ineq}$, equality constraints are defined by $\boldsymbol{C}_{eq}$ and $\boldsymbol{c}_{eq}$. The least square cost function with the form of $\frac{1}{2}||\boldsymbol{A}\boldsymbol{X} - \boldsymbol{b}||^2$ can be expanded:

$$\begin{aligned} \frac{1}{2}||\boldsymbol{A}\boldsymbol{X} - \boldsymbol{b}||^2 &= \frac{1}{2}(\boldsymbol{A}\boldsymbol{X} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{X} - \boldsymbol{b}) \\ &= \frac{1}{2}(\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X} - \boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{A}\boldsymbol{X} + \boldsymbol{b}^T\boldsymbol{b}) \\ &= \frac{1}{2}\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X} - \boldsymbol{b}^T\boldsymbol{A}\boldsymbol{X} + \frac{1}{2}\boldsymbol{b}^T\boldsymbol{b} \end{aligned} \tag{3.13}$$

since $\boldsymbol{b}$ is fixed quantity, it is enough to consider the first two terms, so the cost term can be rewrite into standard QP form $\frac{1}{2}\boldsymbol{X}^T\boldsymbol{Q}\boldsymbol{X} + \boldsymbol{p}^T\boldsymbol{X}$ with $\boldsymbol{Q} = \boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{p} = -\boldsymbol{A}^T\boldsymbol{b}$.

When there are multiple tasks to be optimized, each of them is defined by $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$, weight w_i could be added to define the relative importance among potentially conflicting goals. The overall cost function is:

$$\frac{1}{2}\sum_{i=1}^{N_t} w_i||\boldsymbol{A}_i\boldsymbol{X} - \boldsymbol{b}_i||^2 \tag{3.14}$$

where N_t is the number of tasks. The overall task matrix $\boldsymbol{A}$ and task vector $\boldsymbol{b}$ can be concatenated by all task matrices and vectors:

$$\boldsymbol{A} = \begin{bmatrix} w_1\boldsymbol{A}_1 \\ w_2\boldsymbol{A}_2 \\ \cdots \\ w_{N_t}\boldsymbol{A}_{N_t} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} w_1\boldsymbol{b}_1 \\ w_2\boldsymbol{b}_2 \\ \cdots \\ w_{N_t}\boldsymbol{b}_{N_t} \end{bmatrix} \tag{3.15}$$

3.1.3 Tasks of Interest

Centroidal Linear and Angular Momentum Task

As explained in 2.1.5, the centroidal momentum $\boldsymbol{h}_G$ is the sum of all body spatial momenta computed with respect to the CoM. It collects the system linear momentum and angular momentum. The linear part is directly related to the CoM position with a mass scale. The angular part is not directly related to any orientation representation of the robot [70] but there are evidences showing its usefulnesses in whole-body control, such as resulted upper-body motions and increased robustness to pushes [85] [61] [87]. The centroidal dynamics is given by the equation:

$$\dot{\boldsymbol{h}}_G = \boldsymbol{A}_G\ddot{\boldsymbol{q}} + \dot{\boldsymbol{A}}_G\dot{\boldsymbol{q}} \tag{3.16}$$

where $\boldsymbol{A}_G$ and $\dot{\boldsymbol{A}}_G\dot{\boldsymbol{q}}$ are the CMM matrix and velocity-dependent bias term, they can be constructed from joint space inertia matrix $\boldsymbol{H}$ and velocity term $\boldsymbol{C}\dot{\boldsymbol{q}}$ as described in 2.1.5.

The commanded centroidal dynamics $\dot{\boldsymbol{h}}_{G,c} = [\dot{\boldsymbol{k}}_{G,c}^T, \dot{\boldsymbol{l}}_{G,c}^T]^T$ are composed of the linear part $\dot{\boldsymbol{l}}_{G,c}$ and angular part $\dot{\boldsymbol{k}}_{G,c}$, they are defined separately. For the linear momentum, the commanded rate of change is given by

$$\dot{\boldsymbol{l}}_{G,c} = m[\ddot{\boldsymbol{p}}_{G,d} + \boldsymbol{K}_{D,l}(\dot{\boldsymbol{p}}_{G,d} - \dot{\boldsymbol{p}}_G) + \boldsymbol{K}_{P,l}(\boldsymbol{p}_{G,d} - \boldsymbol{p}_G)] \tag{3.17}$$

where $\ddot{\boldsymbol{p}}_{G,d}, \dot{\boldsymbol{p}}_{G,d}, \boldsymbol{p}_{G,d}$ are desired CoM dynamics comes from trajectories generated by upper-level simple model based controller. $\dot{\boldsymbol{p}}_G$ and $\boldsymbol{p}_G$ are current CoM velocity and position calculated from current robot states. $\boldsymbol{K}_{P,l}$,$\boldsymbol{K}_{D,l}$ are PD gains that could be tuned to achieve desired performance. m is the total mass of the robot.

The commanded angular momentum is given by

$$\dot{\boldsymbol{k}}_{G,c} = \dot{\boldsymbol{k}}_{G,d} + \boldsymbol{K}_{D,k}(\boldsymbol{k}_{G,d} - \boldsymbol{k}_G) \tag{3.18}$$

where $\dot{\boldsymbol{k}}_{G,d}$ and $\boldsymbol{k}_{G,d}$ are desired changing rate of angular momentum and angular momentum, for balancing case they can be set to zero $\dot{\boldsymbol{k}}_{G,d} = \boldsymbol{k}_{G,d} = 0$ which will damp out the system angular momentum. For more dynamic motion, $\dot{\boldsymbol{k}}_{G,d}, \boldsymbol{k}_{G,d}$ could be designed to track some desired angular momentum trajectory to encode whole-body rotation [73].

Foot Pose Tracking Task

Tracking Cartesian space pose targets is considered to be an important task either for manipulation or locomotion. This task also has linear and angular part involved. Starting from the kinematic relationship of an foot:

$$\mathbf{v} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.19}$$

where $\mathbf{v} = [\boldsymbol{\omega}^T, \dot{\boldsymbol{p}}^T]^T$ is the spatial velocity of the foot and $\boldsymbol{J}$ is the associated Jacobian matrix. The second order dynamics can be derived by differentiation:

$$\dot{\mathbf{v}} = \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\ddot{\boldsymbol{q}} \tag{3.20}$$

where $\dot{\mathbf{v}} = [\dot{\boldsymbol{\omega}}^T, \ddot{\boldsymbol{p}}^T]^T$ is the spatial acceleration of the foot. The commanded acceleration $\dot{\mathbf{v}}_c = \left[\dot{\boldsymbol{\omega}}_c^T, \ddot{\boldsymbol{p}}_c^T\right]^T$ are defined using position/orientation PD feedback law:

$$\begin{aligned} \dot{\boldsymbol{\omega}}_c &= \dot{\boldsymbol{\omega}}_d + \boldsymbol{K}_{D,\omega}\left(\boldsymbol{\omega}_d - \boldsymbol{\omega}\right) + \boldsymbol{K}_{P,\omega}\boldsymbol{e}_o \\ \ddot{\boldsymbol{p}}_c &= \ddot{\boldsymbol{p}}_d + \boldsymbol{K}_{D,p}\left(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}\right) + \boldsymbol{K}_{P,p}\left(\boldsymbol{p}_d - \boldsymbol{p}\right) \end{aligned} \tag{3.21}$$

where subscript $[\cdot]_c$ and $[\cdot]_d$ denote commanded and desired quantities. Desired quantities are normally comes from the pose (position and orientation) trajectories given by upper-level planner. Note that $\boldsymbol{e}_o$ stands for orientation error between the desired one and the current one, depending on which orientation representation been used, the calculation differs.

Here, angle-axis representations have been adopted to find the orientation error $\boldsymbol{e}_o$. Given a desired orientation $\boldsymbol{R}_d$ and an actual orientation $\boldsymbol{R}$, the error between them could be defined with an angle-axis rotation:

$$\exp(\theta\hat{\boldsymbol{e}})\boldsymbol{R} = \boldsymbol{R}_d \tag{3.22}$$

where $\boldsymbol{e}$ is a unit vector that defines the direction of the rotation axis and θ is the rotation angle. $\hat{\boldsymbol{e}}$ represents the skew-symmetry matrix corresponding to vector $\boldsymbol{e}$. Together, the exponential map $\exp(\theta\hat{\boldsymbol{e}}) \in SO(3)$ gives a rotation matrix that could rotates the current frame orientation $\boldsymbol{R}$ to desired frame orientation $\boldsymbol{R}_d$ with respect to the inertia frame. They can be solved from elements of the rotation matrix $\boldsymbol{R}_e = \boldsymbol{R}_d\boldsymbol{R}^T$[88]:

$$\begin{aligned} \theta &= \cos^{-1}\left(\tfrac{r_{11}+r_{22}+r_{33}-1}{2}\right) \\ \boldsymbol{e} &= \tfrac{1}{2\sin\theta}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix} \end{aligned} \tag{3.23}$$

The orientation error $\boldsymbol{e}_o$ is then defined as:

$$\boldsymbol{e}_o = \theta\boldsymbol{e} \tag{3.24}$$

where the rotation angle satisfies the condition $||\theta|| \leq \pi$.

To avoid orientation singularities, quaternion representation is often preferred. Given a desired quaternions $\boldsymbol{\epsilon}_d = \epsilon_{d,0} + \epsilon_{d,1}i + \epsilon_{d,2}j + \epsilon_{d,3}k$ and an actual orientation

$\boldsymbol{\epsilon} = \epsilon_0 + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k$, they could be converted back to equivalent rotation matrices $\boldsymbol{R}_d$ and $\boldsymbol{R}$. A unit quaternion $\boldsymbol{\epsilon}_e$ could be extracted from the rotation matrix $\boldsymbol{R}_e = \boldsymbol{R}_d \boldsymbol{R}^T$. The orientation error $\boldsymbol{e}_o$ could be defined as [89]:

$$\boldsymbol{e}_o = 2\epsilon_{e,0} \begin{bmatrix} \epsilon_{e,1} \\ \epsilon_{e,2} \\ \epsilon_{e,3} \end{bmatrix} \tag{3.25}$$

where $\boldsymbol{\epsilon}_e = \epsilon_{e,0} + \epsilon_{e,1} i + \epsilon_{e,2} j + \epsilon_{e,3} k$, in fact, this unit quaternion is related to the angle-axis representation $\epsilon_{e,0} = cos(\frac{\theta}{2})$, $[\epsilon_{e,1}, \epsilon_{e,2}, \epsilon_{e,3}]^T = sin(\frac{\theta}{2})\boldsymbol{e}$, the orientation error $\boldsymbol{e}_o$ in this case is:

$$\boldsymbol{e}_o = 2cos(\frac{\theta}{2})sin(\frac{\theta}{2})\boldsymbol{e} = sin(\theta)\boldsymbol{e} \tag{3.26}$$

To avoid converting the quaternions to orientation matrices, the error $\boldsymbol{e}_o$ can be also directly defined based on $\boldsymbol{\epsilon}_d$ and $\boldsymbol{\epsilon}$ [90]:

$$\boldsymbol{e}_o = \epsilon_0 \begin{bmatrix} \epsilon_{d,1} \\ \epsilon_{d,2} \\ \epsilon_{d,3} \end{bmatrix} - \epsilon_{d,0} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} - \begin{bmatrix} \epsilon_{d,1} \\ \epsilon_{d,2} \\ \epsilon_{d,3} \end{bmatrix} \times \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} \tag{3.27}$$

where

$$\begin{bmatrix} \epsilon_{d,1} \\ \epsilon_{d,2} \\ \epsilon_{d,3} \end{bmatrix} \times = \begin{bmatrix} 0 & -\epsilon_{d,3} & \epsilon_{d,2} \\ \epsilon_{d,3} & 0 & -\epsilon_{d,1} \\ -\epsilon_{d,2} & \epsilon_{d,1} & 0 \end{bmatrix} \tag{3.28}$$

Foot Damping Task

The previous section shows how to track a Cartesian space pose trajectories for robot feet. However, the tracking task is only defined when the foot is not in contact with the ground or its surroundings. When the foot is in contact, it is desired that the foot would be able to adapt to the ground surface and also without drifting. This could be achieved by setting the $\boldsymbol{K}_P$ terms to zeros and gives zero desired velocities and accelerations in (3.21):

$$\begin{aligned} \dot{\boldsymbol{\omega}}_c &= -\boldsymbol{K}_{D,\omega}\boldsymbol{\omega} \\ \ddot{\boldsymbol{p}}_c &= -\boldsymbol{K}_{D,p}\dot{\boldsymbol{p}} \end{aligned} \tag{3.29}$$

In such a way, the foot would behave in a velocity damped way. No knowledge of the terrain is needed and it provides a way to handle uneven terrain scenarios. A good example can be found in [59], a humanoid robot jumps onto the unknown uneven terrain and automatically adapts to it.

Foot Hybrid Force/Motion Task

In certain cases, we would like to simultaneously control the contact forces in vertical direction and the motions in horizontal plane. For example the “moonwalk” dance

movement in which the dancer moves backwards while seemingly walking forwards. Both of the feet are maintaining contact with ground, but one of them is sliding on the surface. For the sliding foot, GRFs in z direction need to be maintained to help whole-body balance while the sliding motion can be imposed on x and y directions by setting the same PD tracking control as before:

$$\begin{aligned}
\ddot{p}_c^x &= \ddot{p}_d^x + K_{D,p}^x \left(\dot{p}_d^x - \dot{p}^x\right) + K_{P,p}^x \left(p_d^x - p^x\right) \\
\ddot{p}_c^y &= \ddot{p}_d^y + K_{D,p}^y \left(\dot{p}_d^y - \dot{p}^y\right) + K_{P,p}^y \left(p_d^y - p^y\right) \\
\ddot{p}_c^z &= -K_{D,p}^z \dot{p}^z
\end{aligned} \tag{3.30}$$

Configuration Task

The configuration of the robot can be direct regulated, for a specific revolute joint i, the PD feedback is defined as:

$$\ddot{q}_{i,c} = \ddot{q}_{i,d} + K_{D,i} \left(\dot{q}_{i,d} - \dot{q}_i\right) + K_{P,i} \left(q_{i,d} - q_i\right) \tag{3.31}$$

It is also possible to directly regulate the orientation of a specific link in Cartesian space. For example the floating base, we could directly apply the orientation part of (3.21) to drive the base to the desired orientation. The Jacobian matrix associated with the floating base depends on the definition of generalized velocity $\dot{\boldsymbol{q}}_1$.

3.2 Applications on Different Contact Situations

In this section, the previously described whole body control framework has been tested in different scenarios. These scenarios differ from each other based on their type of contacts with the environments. Three specific scenarios have been considered here: unknown uneven terrain, prepared structured environments and moving platform.

3.2.1 Unknown Uneven Terrain

The proposed whole-body controller is initially defined for level terrain, in this part it has been demonstrated that the same framework could be adapted to uneven terrain with minor tweaks. An obstacle with with curved surface has been placed under the left foot of the robot. The robot will be released from the air and eventually land on the unknown terrain. After releasing, the robot directly enters the flight phase in which the CoM is not controllable and the robot basically free falls with gravity. However, the relative position of the feet with respect to the CoM is controllable, they have been commanded to stay still below the CoM with a constant offset. To avoid large impact at the landing moment, the positional gains of (3.21) needs to be tuned to a proper value. In such a case, a virtual spring damper system has been simulated between the feet and the CoM. After the landing has been detected, the foot tracking task will be switched to damping mode as described in (3.29). The detecting of landing could be achieved in many different ways, such as checking the z components of the GRFs. It

is easier in simulation since the contact states between two rigid bodies is monitored through out the falling process. The whole process is shown in Figure 3.3.

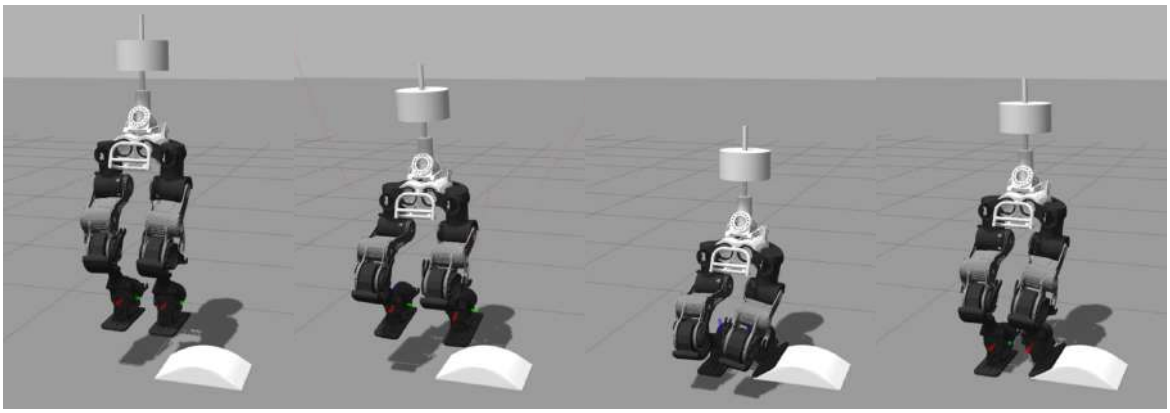


Fig. 3.3 Humanoid robot falls on unknown uneven terrain (from left to right).

3.2.2 Prepared Structured Environments

When a humanoid robot is standing still and has been pushed, it needs to take immediate action to recover from the disturbed states. Depending on the level of perturbation, different strategies could be adopted. For small perturbation, the robot could use its angle torque or moving internal joints to recover [91]. For relatively large perturbation, a step or few steps are needed to recovery [37]. In constrained environment, taking a step might not be possible due to limited space for foothold placement, or the existence of obstacles may completely prevent the stepping, or there is not sufficient time for the swing leg to reach the new foot placement. Nevertheless, there may be situations in which the surrounding environment constraints can be used by hands to obtain supplementary forces in addition to that from the lower limbs. In this case, the robot's balancing capabilities can be extended by exploiting

the constraints of the surrounding environment to get assistance and recover balance under large perturbations. Here, it is particularly emphasized the use of arms and hands for exploring and establishing new contacts with environment.

Environment Setup

To simplify the task, a simple structured environment have been set up to test the proposed control framework. As shown in figure 3.4, two walls have been crossed in a way that they are perpendicular to each other. The robot is standing in front of the conner within a relatively short distance. After been pushed from behind, the robot needs to use the wall as external support to assist balancing. The location of the walls are assumed to be known. They can come from the 3D perception system that is based on vision or point clouds, or a combination of them. The motion planning is needed here to coordinate the whole-body motion after been disturbed, especially the movements of hands and CoM.

Multi-Contacts Motion Planning

To plan the trajectories for hands and CoM, the initiating time and moving duration as well as position/orientation goals are necessary information that have to be decided before at the beginning. More specifically, a perturbation detector is needed to detect the external push and trigger the planning if unavoidable fall is going to happen without taking any action. According to the perturbation direction, wall(s) that assists more on achieving balancing needs to be decided and chosen as the target wall(s). Once the

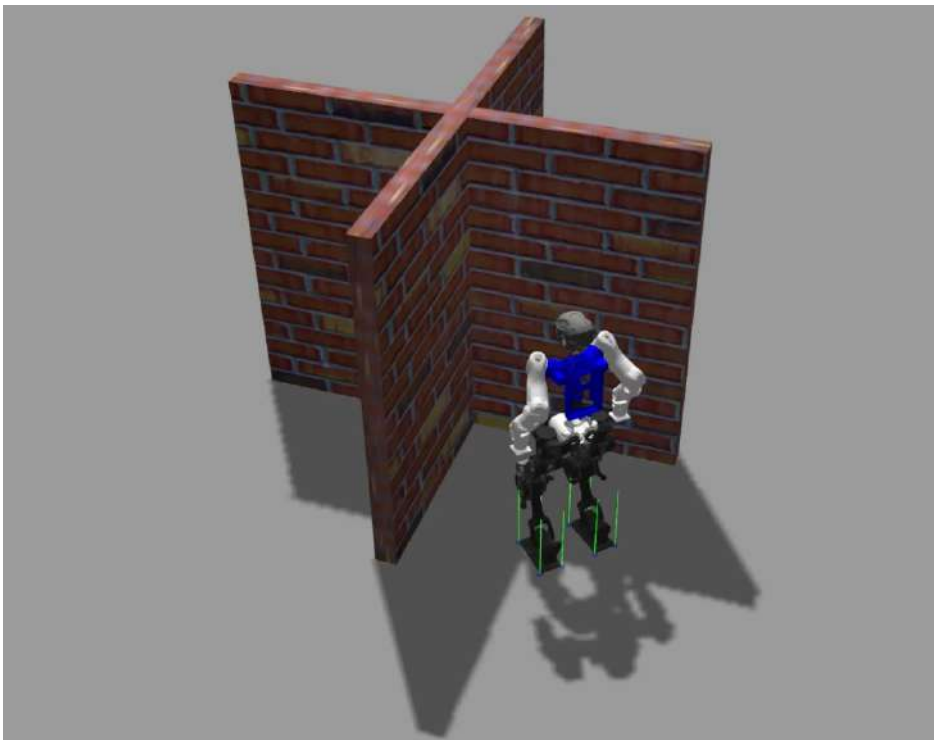


Fig. 3.4 Humanoid robot extends balancing capability by exploiting the wall.

wall(or walls) has been chosen, another module is needed to decide the goal position of the CoM and the goal poses for the two hands (upper body posture helps a lot on the positioning of hands, so it also has been included in the module). The last step of the planning is to generate trajectories based on informations given above. The whole precess has been listed in a workflow chart as shown in Figure 3.5.

Pertubation Detection Many stability criteria could be used as indicators of perturbation, such as ZMP (CoP), CP, Foot Rotation Indicator (FRI), and so on. Here, we choose CoM velocity as the detection indicator. A threshold for CoM velocity magnitude has been set. Once the magnitude goes beyond the threshold, the planner

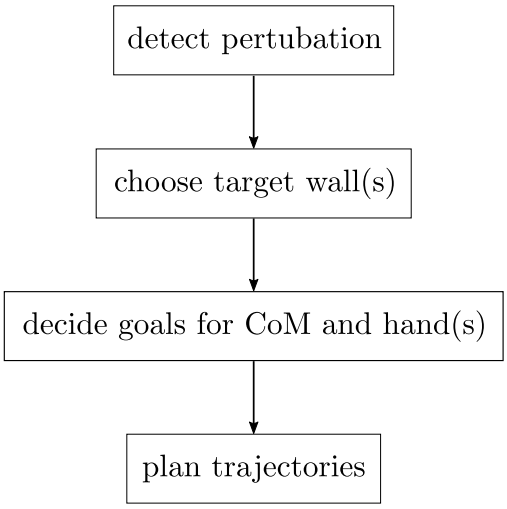


Fig. 3.5 Workflow of the planner.

will be triggered. Not only the magnitude, the direction of the velocity has also been recorded since it is important for later decisions on which wall to push against.

Target Wall(s) Based on the direction of the CoM velocity, it is straight forward to choose the wall in the velocity direction as the target one to assist balancing since it provides reaction forces closer to the reverse velocity direction which helps more on reducing the velocity. Considering the walls in front of the robot has been set perpendicular to each other, multiple choices exist. To simplify the problem, it has been decided that only one hand is allowed to make contact with one wall. That is to say, we do not consider the case in which two hands of the robot push against the same wall. With this assumption, the problem is much simpler. Two yaw angle (calculated from velocity direction) $-30\,\text{deg}$ and $30\,\text{deg}$ has been chosen as thresholds. If the yaw angle is between the area of $[-30, 30]\,\text{deg}$, two walls will be selected as target walls, two hands of the robot will be used and each of them will make contact with the

closest one next to it. If the yaw angle gose beyond this range, the wall in the velocity direction will be chosen as target wall and the robot will only use one hand to balance. These rules are illustrated in Figure 3.6.

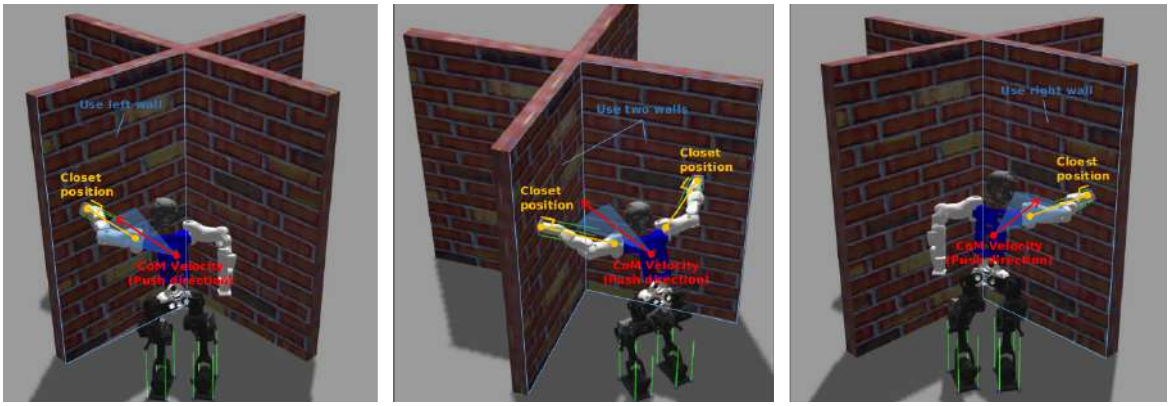


Fig. 3.6 Choosing target wall(s) according to the perturbation direction.

Goals of Hand(s) After the target wall has been decided, hands involved also fixed. On the target wall, the closest point to the relevant shoulder will be selected as the contact position for the corresponding hands. The orientation of the wall is used as the target orientations for contacting hand(s). The poses of hands when perturbation was detected have been used as the initial poses for generating trajectories. Interpolating the initial and target poses, we can get the trajectories of hands. For the positional part, quintic polynomial has been used. Slerp has been used to interpolate the orientation quaternions.

Goals of CoM Another important part of the high level planner is the generation of CoM trajectory. The key issue for establishing stable contact between the hands and the wall is to synchronize the motion of the CoM and hands. Without actively

modulating the CoM, the robot is passively falling forward under a strong perturbation, which is not desirable for the control of the motion of hands. The idea is to generate a controlled fall motion that can be planned at the moment when the robot been disturbed. With actively controlled fall, the ZMP (CoP) can be regulated to guarantee that the feet stays on the ground throughout the whole process. Here, Linear Inverted Pendulum Model (LIPM) has been chosen as the template model to generate the CoM motion. Its dynamics in x direction can be described as:

$$\ddot{x} = \frac{g}{z_0}(x - x_f) \tag{3.32}$$

where x and x_f refer to CoM and foot position, z_0 is the CoM height and g is the gravity constant. It can be solved in time domain the analytical solution is:

$$x(t) = Ae^{-t/T_c} + Be^{t/T_c} + x_f \tag{3.33}$$

where parameters are defined by the model constants and initial conditions:

$$\begin{aligned} T_c &= \sqrt{z_0/g} \\ A &= (-T_c\dot{x}(0) + x(0) - x_f)/2 \\ B &= (T_c\dot{x}(0) + x(0) - x_f)/2 \end{aligned} \tag{3.34}$$

With this formula, we can generate the trajectory for CoM given its initial state and the foot position when perturbation been detected. Based on a decided fall motion period T, the planner will synchronize the trajectories of the hands and CoM. The

activated hands are expected to reach desired location within the given period. For the CoM, the LIPM will be used to simulate forward for the same period T and generate the trajectory during this period. These trajectories are all planned at the perturbations detected moment and will be sent to the whole-body controller throughout the ensuing time period T.

Switch Control Mode After all these trajectories been accomplished, the planner will change its control mode: from tracking model to balancing model. At the moment of T, all end-effector tracking tasks will be switched to damping mode. The CoM tracking task will be changed to regulation task which will regulate the CoM position to the last point of previously given trajectory.

Recovery Motion Planning

For the recovery motion (the robot recovers from wall supporting posture to standing posture), it is basically an reversed process of the falling one. The robot needs to move its hands and CoM back to their initial locations when standing still on the ground. It is obviously impossible to move the CoM and supporting hands together starting from that static supporting posture since the CoM is located outside of the support polygon form by its two feet, suddenly removing the supporting hands will cause the robot to fall against the wall and it is also impossible to drag itself back with limited ankle torque. The robot needs to do more dynamic motion to push itself back. The strategy we proposed here is to command the robot first to generate a initial launching velocity backwards and then remove the hands, the rest part is basically let the CoM

go passively to its desired rest position. The orbital energy of LIPM is conserved and can be written as:

$$\frac{1}{2}\dot{x}^2 - \frac{g}{2z}x^2 = \text{ constant } \equiv E \tag{3.35}$$

the launching velocity can be calculated base on the energy at peak point. Assuming the point mass is at the peak point when standing still, then the velocity at given position x that can bring the point mass back to its peak and rest on it can be calculated. After this lunching velocity been found, the analytic solution of LIPM could be applied to plan the reverse CoM trajectory. The whole-body controller will then take care of the hands and CoM trajectories tracking. For this specific recovery motion, the tasks considered in whole-body controller includes: 1) CoM position tracking; 2) Pevlis orientation tracking; 3) Hand pose tracking before touching the wall; 4) Minimize the contact torques and tangential forces for all contact points after the hands touching the wall; 5) Angular momentum regulation.

Contact force distribution should be considered for all contact points. Here, tangential contact forces have been minimized to prevent unstable contacts. Figure 3.7 shows the measured contact forces for for all contact points when the robot is supporting against the wall with both of its hands.

In the whole push recovery scenario, the robot is able to switch freely between two contact points (two feet) and three/four contact points (feet and hands) to make use of external supports to help balancing. Figure 3.8 shows the whole process.

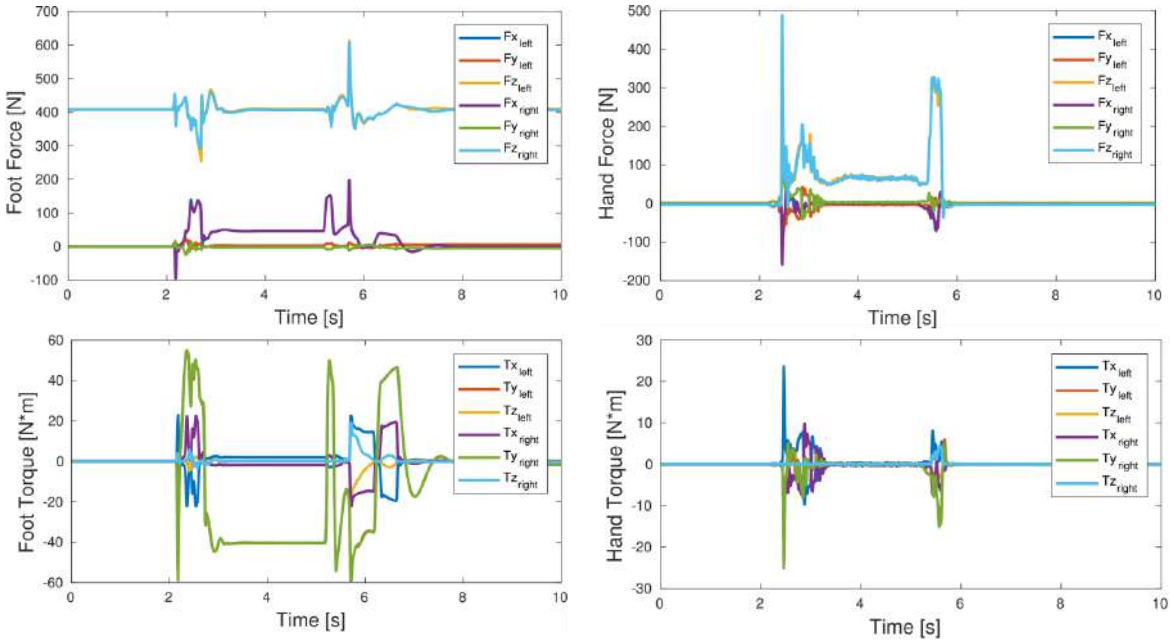


Fig. 3.7 Contact forces at hands and feet contact points.

3.2.3 Moving Platform

Humans are able to perform a wide variety of tasks and humanoid robots are created with the expectation to have comparable capability and versatility [92]. Most daily objects and tools are designed to fit in human size and in such a way can be easily handled and operated. Resembling the human body, humanoid robots can potentially take advantages of it, thereby avoiding the need to alter the environment or modify its own structure. Even though how to use human-oriented tools has been extensively studied in the manipulation tasks performed by robotic arms, few attentions have been drawn to those tools which are supposed to be operated by the lower limb or whole body of human. Wheeled mobile transportation platform is a very important tool of this kind. In general, wheeled platforms consume less energy and move faster

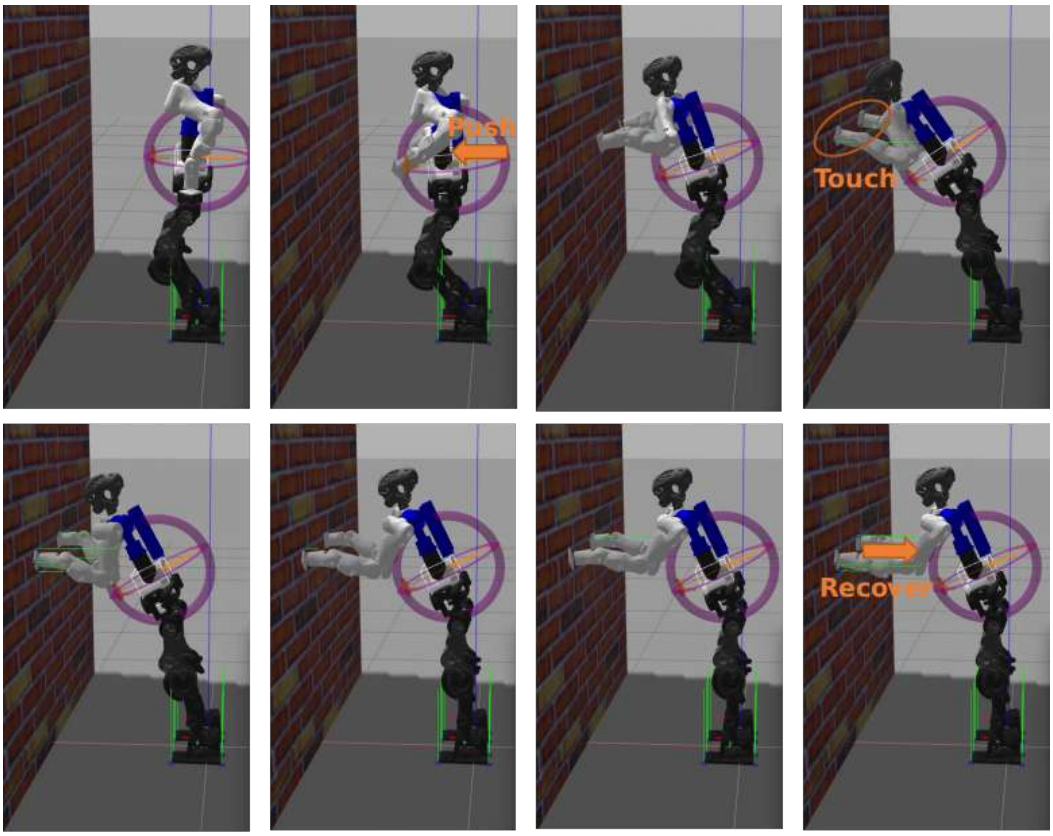


Fig. 3.8 WALK-MAN makes use of a wall in front of itself to balance and recover from a strong push from behind.

than legged robots in terms of mobility, therefore it is necessary to investigate the manoeuvrability of them for humanoid robots.

Comparing to other wheeled mobile platforms, two-wheeled mobile platform (TWMP), well known as Segway™, is more convenient and lightweight. It has the advantages of small footprint, zero turning radius and relatively large carrying payload [93]. The modeling and control of TWMP have also been widely studied. In 2002, the Swiss Federal Institute of Technology built a scaled down prototype of a two-wheeled vehicle, named JOE, which is able to balance itself while tracking commanded velocity inputs [94]. At the same year, Segway™ Personal Transporter was brought to market

Fig. 3.9 COMAN riding on a two-wheeled mobile platform

as a new mobile platform for human transporting [93]. Different control methods were proposed to improve its performance [95–97].

To make use of the mobility of the wheeled platform for the humanoid robot, several attempts have been made. The Johnson Space Center developed a mobile manipulation system in which the upper body of the NASA/DARPA Robonaut system is attached to Segway^{TM} robotic mobility platform yielding a dexterous, maneuverable humanoid [98, 99]. Recently, Boston Dynamics^{TM} released a new robot, Handle, which is a wheel-leg hybrid robotic system that can take advantage of both humanoid robots and wheeled mobile robots. The Handle robot is an integrated system and the control algorithms has to treat it as a whole. In this paper, instead of modifying the humanoid robot, we are going to explore how to use the existing humanoid robot to operate the TWMP without additional hardware customization. More specifically, we attempt to make our humanoid robot COMAN (COmpliant HuMANoid Platform) operate the

two-wheeled mobile platform as shown in Fig. 3.9. Hyungjik et al. have implemented similar idea on their position-controlled humanoid robot [100]. The humanoid robot can lean forward or backward to regulate its center of gravity to control the movement of mobile platform. But since the employed humanoid robot is position-controlled, it is difficult for the humanoid robot to resist large disturbance and perform compliant motions.

COMAN is actuated by passive compliance actuators based on the series elastic actuation principle (SEA) and is capable of being torque-controlled. The whole-body dynamic model of the humanoid robot allows us to calculate required torques for a specific motion considering dynamic coupling effects. The passive compliance actuators can reject small perturbation and make the robot behaves compliantly.

Torque-controlled robots become more and more available and many related algorithms are developed. Passivity-based approaches [101, 102] compute admissible contact force and control commands under quasi-static assumptions without the need of full dynamic model. However, more dynamic motions can be handled by considering the full dynamic model of the robot [51, 103–105]. What is common between these approaches is that they all regulate the position of the center of mass (CoM) of the robot to ensure that the robot does not fall while maintaining the contact forces in physically achievable range. To achieve better performance for balancing the robot, momentum-based controller was proposed [55, 106, 103]. In such approach, both CoM motion (i.e. linear momentum) and its angular momentum are controlled. Optimization methods [85, 107–110] are used as a tool to calculate joint torques based

on whole-body dynamics. Experiments on various robots shows impressive, human-like balancing behaviors [51, 58, 111–113]. Based on the investigations in both research fields, we decided to use the whole-body dynamic torque control strategy to stabilize the humanoid robot on a TWMP and drive it.

Model and Control Strategy of TWMP

The TWMP is actually a mobile inverted pendulum and its model have been widely studied in the field of autonomous robotics [114, 94, 115].

The mobile platform has three degrees of freedom (DoF): 1) the rotation about the the wheel axis, this movement is intrinsically unstable, the body part of the inverted pendulum tends to fall if given no control, 2) the linear movement in the heading direction, 3) the steering rotation which changes the heading of the robot.

The coordinate frame is shown in Fig. 3.10. Three coordinate frames are plotted in the figure: one world frame $\{F_w = \{x_w, y_w, z_w\}$, one intermediate frame $\{F'\} = \{x', y', z'\}$ and a local frame $\{F\} = \{x, y, z\}$. The dynamic of the robot can be fully described with six parameters: θ_P and ω_P stands for the pitch angle and angular velocity around the y axis. The robot position and velocity in the heading direction is defined as x_{RM} and v_{RM}. Additionally, θ_Y and $\dot{\theta}_Y$ are the yaw angle and associated angular velocity around the z_w axis. The nonlinear dynamic model of the inverted pendulum model follows the equations given in [94, 96].

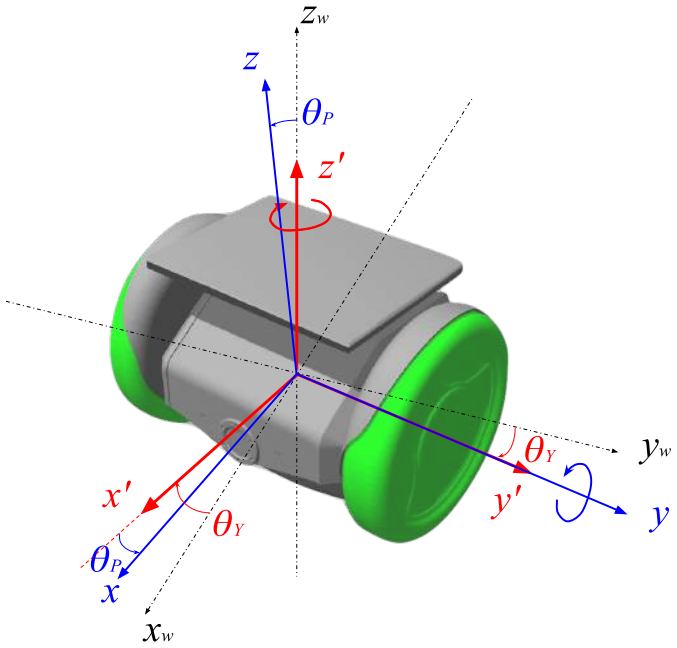


Fig. 3.10 The mobile inverted pendulum model

Linearizing the nonlinear model around the operating point($x_{RM} = 0$, $v_{RM} = 0$, $\theta_P = 0$, $\theta_Y = 0$) the system can be written in state-space form:

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{u} \tag{3.36}$$

where $\boldsymbol{X} = [x_{RM}, v_{RM}, \theta_P, \omega_P, \theta_Y, \dot{\theta}_Y]^T$ denotes the state vector, $\boldsymbol{u} = [C_L, C_R]^T$ are input torques on left and right wheel.

The decoupling transformation developed in [94] decomposes the above system into two independent subsystems.

$$\begin{bmatrix} C_L \\ C_R \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} C_P \\ C_Y \end{bmatrix} \tag{3.37}$$

One subsystem relates to the rotation and linear translation in sagittal plane:

$$\begin{bmatrix} \dot{x}_{RM} \\ \dot{v}_{RM} \\ \dot{\theta}_P \\ \dot{\omega}_P \end{bmatrix} = \boldsymbol{A}_P \begin{bmatrix} x_{RM} \\ v_{RM} \\ \theta_P \\ \omega_P \end{bmatrix} + \boldsymbol{B}_P C_P \tag{3.38}$$

The other subsystem describes the steering of the mobile robot in transverse plane:

$$\begin{bmatrix} \dot{\theta}_Y \\ \ddot{\theta}_Y \end{bmatrix} = \boldsymbol{A}_Y \begin{bmatrix} \theta_Y \\ \dot{\theta}_Y \end{bmatrix} + \boldsymbol{B}_Y C_Y \tag{3.39}$$

Then two independent controllers can be designed for each subsystem. For the sagittal plane inverted pendulum subsystem, the control goal is to achieve self-balancing without falling down. The yaw control goal is simply to regulate the turning rate to a desired value.

The design of the state-space feedback controllers follows textbook approaches which formulate a stable close-loop controller to drive the system state to the desired values.

The TWMP provides the user a command interface through which the user is able to send desired forward speeds and steering rates to control the platform directly.

Simulation

Several tasks are conducted to verify previously proposed control scheme. The humanoid robot used here is COMAN [67] and its full dynamic model is used in the simulation.

COMAN body has 29 DoF in total: 6 DoF for each leg, 3 DoF waist and 7 DoF for each arm. In the simulation, each joint is torque controlled and the joint level controller is a combination of feed-forward term and feedback term:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{des}} + \boldsymbol{K}_p(\boldsymbol{q}_{\text{des}} - \boldsymbol{q}) + \boldsymbol{K}_d(\dot{\boldsymbol{q}}_{\text{des}} - \dot{\boldsymbol{q}}) + \boldsymbol{K}_i \int (\boldsymbol{q}_{\text{des}} - \boldsymbol{q}) \tag{3.40}$$

Where $\boldsymbol{\tau}_{\text{des}}$ is the desired joint torque computed from inverse dynamics controller, $\boldsymbol{q}_{\text{des}}$ and $\dot{\boldsymbol{q}}_{\text{des}}$ are the desired joint position and velocity integrated from the desire joint acceleration which is the part of the optimization variable. $\boldsymbol{K}_p$, $\boldsymbol{K}_d$ and $\boldsymbol{K}_i$ are PID gains for the feedback term. In the simulation, we don't use the feedback part and merely set feedback gains to zero. But the feed-back part is important in real robotic system considering modelling error and sensor noises. In those cases, the feed-forward torque dominates the control command while feedback torque are mainly used to stabilize the joint. The control frequency is 500 Hz for the humanoid.

The TWMP used in the simulation is the open source RoboSavvy$^{\text{TM}}$ self-balancing robotic platform [116]. Since provided a velocity command interface, we can send a pre-defined velocity profile to the platform and make it as test platform which could be used to test the stability of the humanoid robot. The first task is to make the humanoid robot act as a camera stabiliser. The second task is to let the humanoid robot drive the mobile platform to a desired location. In this task, no velocity command will be sent to the mobile platform.

Task: Balancing and Camera Stabilizing The primary problem for COMAN riding on the mobile platform is to guarantee the stability when standing on the platform. It would not be a difficult task since the platform can stably carry the rider with its own controller. Even with certain amount of external disturbance and payload variation, it could work as well. Therefore, we would like to assign additional tasks to COMAN. In this simulation, other than merely standing on the platform, COMAN was also expected to act as a stabilizer for the camera mounted in its head in order to capture steady images. Note that COMAN and the mobile platform are two separated system without knowing the control details of each other. For COMAN, it will treat the movement of the platform as external disturbance and should be able to cope with it properly. For the mobile platform, it will treat the movement of COMAN as disturbance as well.

For balancing of the humanoid robot, whole-body dynamics should be utilized to regulate the linear and angular momentum of the whole system. The desired linear and angular momentum $\boldsymbol{P}_{\text{des}}$, $\boldsymbol{L}_{\text{des}}$ and their changing rates $\dot{\boldsymbol{P}}_{\text{des}}$, $\dot{\boldsymbol{L}}_{\text{des}}$ were set zero. And the desired CoM position $\boldsymbol{C}_{des}$ in this task was given as: $\boldsymbol{C}^{x,y}_{\text{des}} = \frac{1}{2}(\boldsymbol{P}^{x,y}_{L} + \boldsymbol{P}^{x,y}_{R})$, $\boldsymbol{C}^{z}_{\text{des}} = z_c$. $\boldsymbol{P}_L$ and $\boldsymbol{P}_R$ are the locations of the two feet in the world frame. z_c is given constant CoM height, and it should be within the kinematic limits of the robot. The superscripts indicate the corresponding components. The x and y components of the desired CoM position were equal to the geometrical center of two feet, and the z component was set to be constant with respect to the ground frame. The consideration behind this was: we would like to keep the ground projection of CoM as far as possible

away form the boundary of the feet. In addition, to stabilize the internal camera, CoM should not oscillate too much in the vertical direction with respect to the ground frame. In real system, these global references would be given by the localization system. And the image captured from the camera could be used as feedback to decide the reference height.

The camera is installed in the head of COMAN, which is relatively fixed with respect to the torso. To stabilize the camera means to control the torso orientation. The desired orientation given here is identical to the ground frame. And the corresponding angular velocities and accelerations are zero. $\boldsymbol{R}_{\text{des}} = \boldsymbol{I}_3$.

Desired linear velocity $v = A\sin(2\pi ft + \phi)$ in heading direction and turning rate $\omega = 0$ were sent to the mobile platform individually. t is time, A is amplitude, f is the ordinary frequency and ϕ is the phase at $t = 0$. Following the sine wave linear velocity and zero turning rate, the mobile platform will move forward and backward in sagital plane and evoke disturbance to the standing of COMAN.

To evaluate the balancing ability of COMAN, other than checking the fluctuation of torso orientation, we would like also to have an intuitive feeling by just comparing the images collected separately from the two cameras mounted on COMAN and the mobile platform. As shown in Figure 3.11, the two cameras were close to each other and both shot towards the wall. The performance of COMAN stabilizing the camera was very impressive as seen in Figure 3.12. The thumbnails in the above row comes from the camera of the mobile platform which vibrated a lot. On the contrary, the

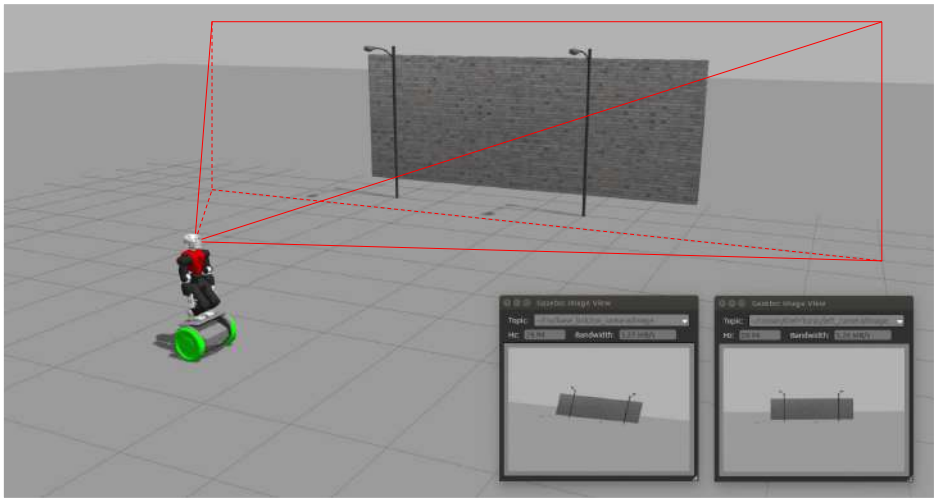


Fig. 3.11 The simulation setup. At the right bottom, views of the cameras are displayed: the left one is from the camera installed on the TWMP and the right one is from the one in the head of COMAN.

ones from COMAN's camera shown below were much more stable. The time interval between these images was 1.5 seconds for both cameras.

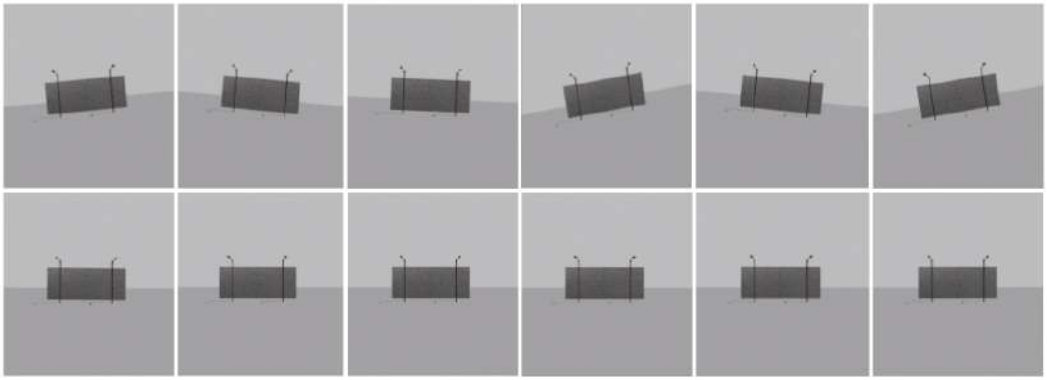


Fig. 3.12 Comparison between camera images (interval 1.5 seconds, the above row came from the camera mounted on the TWMP and below was form the one in the head of COMAN).

The pitch angles of COMAN toros and the TWMP were shown in Fig.3.13. The varying range of pitch angle of COMAN torso was approximately 10 persent of the one measured from the TWMP, which was a large improvement.

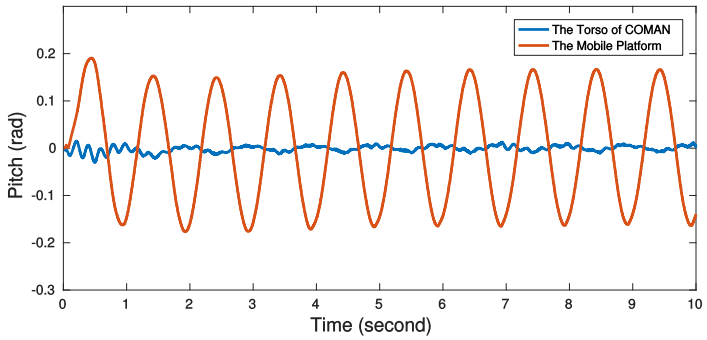


Fig. 3.13 Comparison between pitch angles of COMAN toros and the TWMP

Task: Riding the TWMP One fantastic thing of the TWMP is that human riders can head to desired directions by leaning their bodies. To imitate this skill, COMAN was controlled to shift its CoM position forward and backward to regulate the forward velocity of the TWMP (see Figure 3.14). For turning, there are different kinds of devices for human rider to send the steering command, such as handlebar or twisting pedal. To be consistent with the way of regulating forward velocity, here we detect the force distribution on the left and right wheels caused by shifting COMAN body left and right. According to the distribution, a steering command was generated by the TWMP and then it would turn to the direction that COMAN wished to go.

To testify the feasibility of the aforementioned control strategy, we commanded COMAN to drive the TWMP to a desired pose $\boldsymbol{P}_{des} = [x, y, \theta]^T$ which was given here as $\boldsymbol{P}_{des} = [2, 2, \pi/4]^T$ expressed in global frame. COMAN started from a initial pose $\boldsymbol{P}_{ini} = [0, 0, 0]^T$ and drove to the goal pose by shifting its CoM.

The tracking error used to generate CoM offset of COMAN was defined as $\boldsymbol{e} = [e_x, e_y, e_\theta]^T = \boldsymbol{P}_{\text{des}} - \boldsymbol{P}_{\text{cur}}$, and $\boldsymbol{P}_{\text{cur}}$ was the current pose of the TWMP. The control law was as: $\Delta x = K_x\sqrt{e_x^2 + e_y^2}$, $\Delta y = K_y e_y + K_\theta e_\theta$ where Δx and Δy were CoM shifts

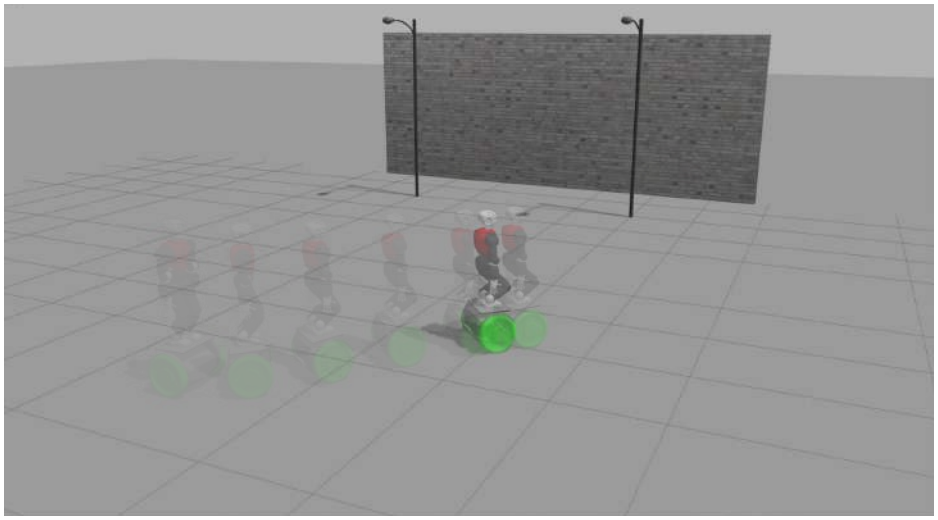
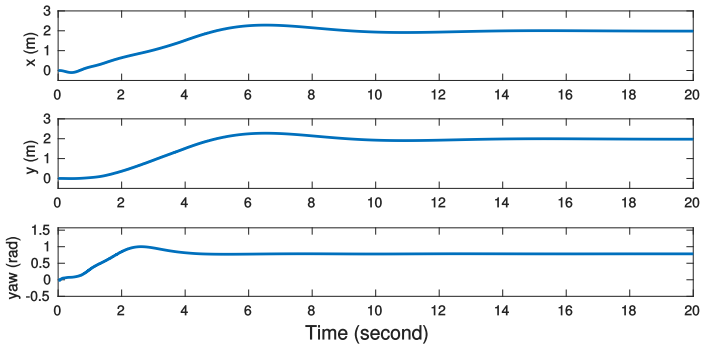


Fig. 3.14 COMAN drove the TWMP from the initial pose $[0, 0, 0]^T$ to the goal pose $[2, 2, \pi/4]^T$.

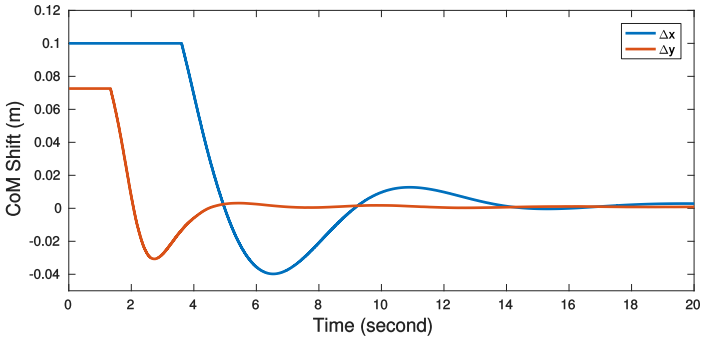
from the center of the feet in the foot local frame. K_x, K_y and K_θ were feedback gains. Δx was in the forward direction which would affect the forward velocity while Δy was along lateral direction and related to the turning rate. To be noted, the calculated COM shifts would be truncated if they were out of the support polygon.

Apart from reaching to a goal pose, we also expected the torso of COMAN to be upright and head forward with respect to the TWMP all the way. Taking the turning of the TWMP into consideration, the desired orientation of COMAN was defined as: $\boldsymbol{R}_{\text{des}} = \boldsymbol{R}_z(\boldsymbol{P}^{\theta}_{\text{cur}})$ where $\boldsymbol{R}_z(\theta)$ was an elemental rotation matrix that rotates a vector by an angle θ about the z axis. $\boldsymbol{P}^{\theta}_{\text{cur}}$ is the current orientation of the TWMP.

The tracking data of the TWMP pose in this simulation is given in Fig. 3.15a. COMAN successfully drove the TWMP to the goal location. And the corresponding CoM shift is shown in Figure 3.15b. At the beginning, the commands were truncated



(a) The pose of the TWMP. It started from the initial pose $[0, 0, 0]^T$ and reached the final pose $[2, 2, \pi/4]^T$ after about 10 seconds.



(b) The CoM shift of COMAN.

because of the feet size limits. These limits prevent the humanoid robot shifting too much which would result the robot tilting on the TWMP.

Balancing and Camera Stabilizing on Uneven Terrain The previous simulations were performed on flat ground. In this part, we would like to challenge the proposed controller on uneven terrain.

The setup of this simulation was similar with 3.2.3 except that COMAN had to deal with additional disturbances introduced by the terrain (see Figure 3.16). Figure 3.17a shows that the image taken from the TWMP shook a lot and deviated from the target while the camera on COMAN still faced the right direction. The image deviation was because here the yaw regulation of the TWMP had no feedback control and would

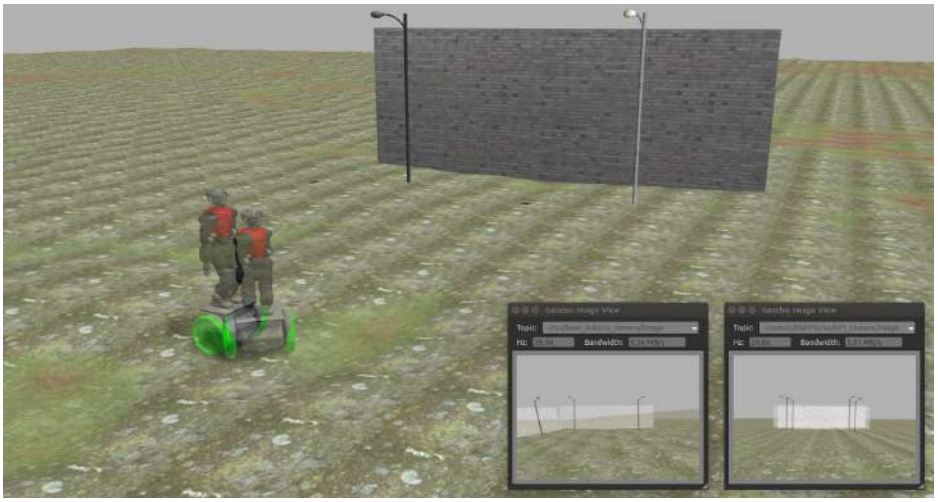
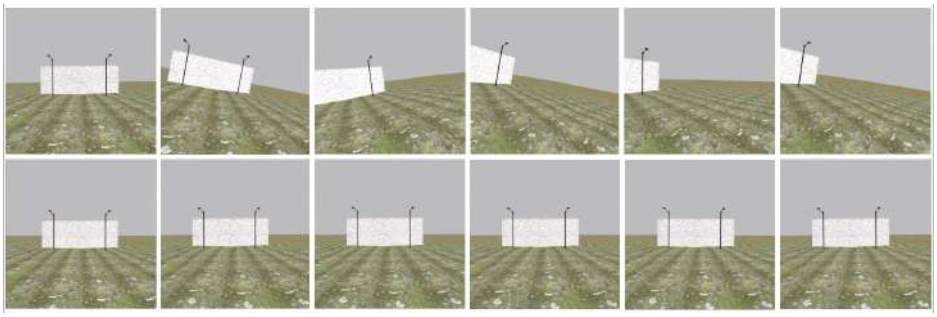


Fig. 3.16 Snapshot of the simulation on uneven terrain

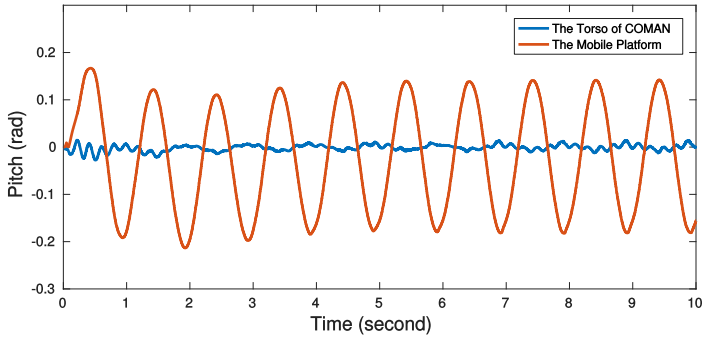
gradually drift away under the disturbance of uneven terrain. The pitch angles of COMAN torso and TWMP are shown in Figure 3.17b. Much smaller fluctuation of COMAN torso was observed compared with the TWMP, similar with the result on flat ground.

Riding the TWMP The initial pose and goal pose set in this simulation were the same with 3.2.3 (see Figure 3.18). The result in Figure 3.19a shows COMAN successfully reached the desired pose. And the corresponding CoM shift is shown in Figure 3.19b. That the CoM shift did not converge to zero at the end is because COMAN needed to resist the inclination of the terrain at the goal location and keep the TWMP staying on the slope.

In this part we controlled the humanoid robot COMAN to perform two different tasks by utilising the transportation tool TWMP. The first one is to stabilize the camera which is installed in its head while balancing on the TWMP. Another task is



(a) Comparison between camera images. The above row comes from the camera mounted on the mobile platform and the below row is taken form the one in the head of COMAN (interval 3 seconds).



(b) Comparison between pitch angles of COMAN torso and the TWMP on uneven terrain

Fig. 3.17 Balancing and Camera Stabilizing on Uneven Terrain.

to drive the TWMP to the desired location. Both tasks are performed on even terrain and uneven terrain.The humanoid robot successfully demonstrated its ability to utilise device designed for human and its versatility and adaptivity to different tasks and environment.

With balancing and locomotion abilities demonstrated in this paper, as a natural extension, we would like to explore more tasks such as going down stairs, object manipulation, cooperation with human co-workers or other robots.

Another issue is the utilization of angular momentum. In these tasks, we simply set the desired angular momentum to zero, which helped to stabilize the body of COMAN. However, we found that it would hinder the operation of the TWMP. When TWMP

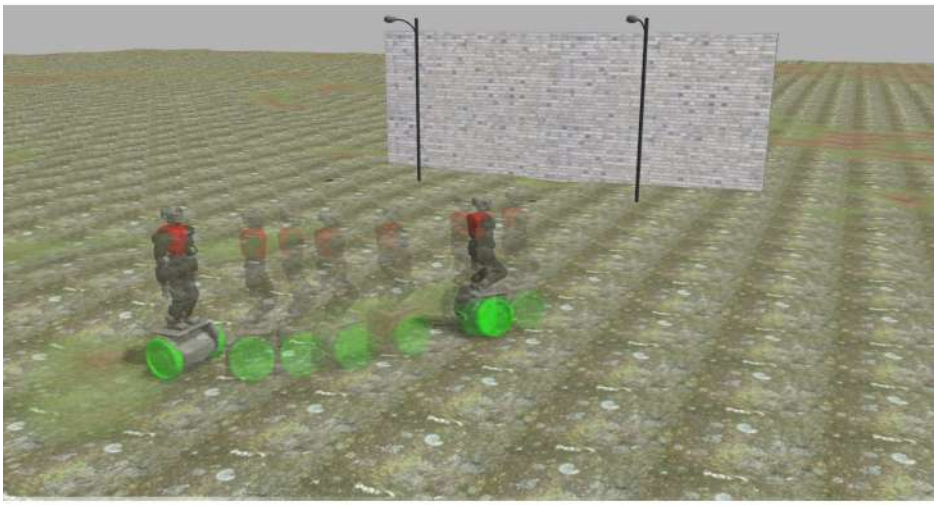
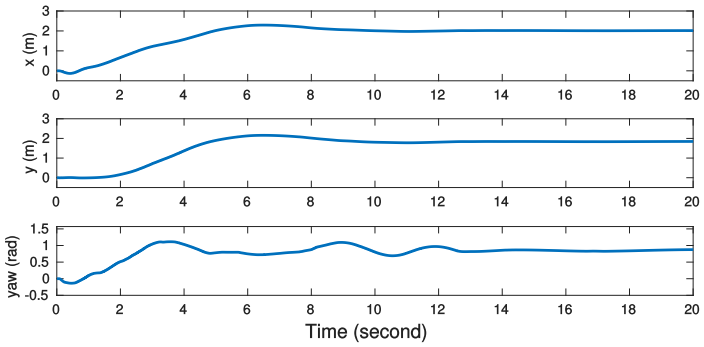


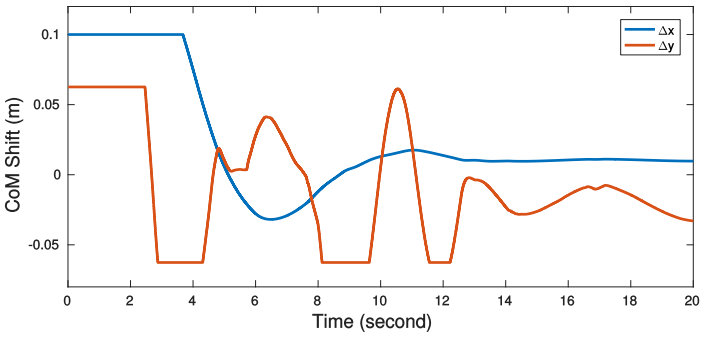
Fig. 3.18 COMAN drove the TWMP from the initial pose $[0, 0, 0]^T$ to the goal pose $[2, 2, \pi/4]^T$ on uneven terrain.

tried to accelerate, it would tend to lean forward which would cause the changing of angular momentum of COMAN. As a result, COMAN would counteract the changing and slow down the acceleration of TWMP. It should be a better choice to define the desired angular momentum of COMAN according to the expected movement of the whole system.

In addition, as shown in the camera stabilizing simulation, this system is a perfect platform for capturing stable image information about the world around it and therefore should serve well for environment mapping and localization of itself.



(a) The pose of the TWMP. It started from the initial pose $[0, 0, 0]^T$ and reached the final pose $[2, 2, \pi/4]^T$ after about 15 seconds.



(b) The CoM shift of COMAN on uneven terrain

Fig. 3.19 Riding the TWMP.

Chapter 4

Walking

Robust walking of humanoid robots often requires the robot to be adaptive to external disturbances and terrain irregularities. Three push recovery strategies that allow the robot to recover from different levels of external push has been presented in [91]: regulating the center of pressure ("ankle strategy"), changing centroidal angular momentum ("hip strategy") and taking a step("stepping strategy"). For small disturbances, "ankle strategy" and "hip strategy" would be enough. As the disturbance increases, "stepping strategy" has to be used to stop from falling. In this part, it will be focused on the continuous stepping strategy since it is very effective for most scenarios. On top of this, two new configurations have been proposed to enable the robot to do cross-step which addresses some limitation of the current stepping strategy. At the end, hopping strategy has also been introduced to enhance the push recovery ability.

4.1 Simple Model Based Predictive Control

4.1.1 Previous Works

Re-planning of gait trajectory is a crucial ability to compensate for external disturbances and it also leads to robust locomotion ability. There are many researches that have been done on using Model Predictive Control [117] for online walking pattern generation and push recovery. Different MPC formulation have been proposed to updates foot placement online [118] [119] [120] [121]. To meet real-time requirement, linear models are often chosen as template model to perform iterative online optimization involved in the MPC control scheme. More specifically, foot placement and CoM trajectory can also be generated simultaneously by solving a linear trajectory optimization problem using LIPM dynamics. In these approaches, foot placement are generated automatically based on some overall behavioral goals such as a desired average speed or reaching some distant position.

In [118], a continuous version of foot placement optimization is proposed in the paper . The paper demonstrated the capability of Model Predictive Control to generate stable walking motions without the use of predefined foot steps. In the paper, the foot placement has been included in optimization problem and it can be generated automatically given a reference speed of the robot. Together with the foot placement, the CoP trajectories are also been optimized. It means that the ankle torque has been used to modulate the CoP position. Here in this thesis, only foot placements are considered, the CoP modulation is included in whole-body controller as a constraints.

In many previous implementation like [120] and [121], the idea of automatic foot placement has already been implemented. However, they all keep some kind of reference foot placement tracking term in their optimization goals.

In the works [120], the term which minimizes the least square errors between the optimized and desired foot placements has been added:

$$\min_{\boldsymbol{p}_i} \sum_{i=1}^{N} ||\boldsymbol{p}_i - \boldsymbol{p}_i^*||^2 \tag{4.1}$$

where $\boldsymbol{p}_i^*$ is a serials of desired foot placements are generated based on the desired average speed $\dot{\boldsymbol{v}}^* = [\dot{x}^*,\ \dot{y}^*]^T$:

$$\boldsymbol{p}_i^* = \boldsymbol{p}_0 + 2iT \times \boldsymbol{v}^* + (i \bmod 2)[0 \quad dk]^T, \quad 1 \leq i \leq N \tag{4.2}$$

where d is inter-feet distance (hip width could be used) and k is 1 for right support phase and -1 for left support phase. In lateral direction, this term is important since without desired foot placement anchoring the foot steps become closer and closer, finally overlap which is not desired due to self collision.

In the work [121], there is a similar term which minimizes the deviation of planed foot placements from the desired foot placements:

$$\min_{\boldsymbol{p}} \sum_{t} \left(\boldsymbol{X}_t - \boldsymbol{X}_t^*\right)^T V_{XX}^t \left(\boldsymbol{X}_t - \boldsymbol{X}_t^*\right) + w \left(\boldsymbol{p} - \boldsymbol{p}^*\right)^2 \tag{4.3}$$

where $\boldsymbol{p}^*$ is a sequence of given desired foot placements, the nominal CoM trajectory $\boldsymbol{X}_t^*$ is also generated based on it.

Actually, the tracking of the reference foot placements can be replaced by just enforcing constraints on the position of foot placements based on the observation from [35]. Here in this dissertation, we propose a different formulations which does not require any reference placements anchoring which makes the planning truly automatic.

4.1.2 Improved Formulation

The formulation is still based on LIPM and its dynamics in x direction is:

$$\ddot{x} = \frac{g}{z_0}(x - x_f) \tag{4.4}$$

where x and x_f refer to CoM and foot position, z_0 is the CoM height and g is the gravity constant. If we collect the position and velocity of CoM as a state $\boldsymbol{x} = [x,\ \dot{x}]^T$, the state in time t can be calculated as:

$$\boldsymbol{x} = \boldsymbol{A}(t)\boldsymbol{x}_0 + \boldsymbol{B}(t)x_f \tag{4.5}$$

where $\boldsymbol{x}_0$ is the initial state, $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$ are time dependent matrices:

$$\begin{aligned} \boldsymbol{A}(t) &= 0.5 \begin{bmatrix} e^{\omega t} + e^{-\omega t} & \frac{e^{\omega t} - e^{-\omega t}}{\omega} \\ \omega \left(e^{\omega t} - e^{-\omega t}\right) & e^{\omega t} + e^{-\omega t} \end{bmatrix} \\ \boldsymbol{B}(t) &= \begin{bmatrix} 1 - 0.5 \left(e^{\omega t} + e^{-\omega t}\right) \\ 0.5\omega \left(e^{-\omega t} - e^{\omega t}\right) \end{bmatrix} \end{aligned} \tag{4.6}$$

where $w = \sqrt{g/z_0}$.

It is assumed there is no double support phase involved in the formulation. The robot switches between left support and right support immediately. Fixed step time T also been assumed to keep the formulation linear. During swing phase, given the remaining duration of the current swing phase t_{TD}, the current support foot position $x_{f,0}$ and current estimated CoM state $\boldsymbol{x} = [x, \dot{x}]^T$, the CoM state at touch down moment $\boldsymbol{x}_0$ could be calculated:

$$\boldsymbol{x}_0 = \boldsymbol{A}(t_{TD})\boldsymbol{x} + \boldsymbol{B}(t_{TD})x_{f,0} \tag{4.7}$$

Since no double support phase assumed, the take off state is the same as touch down state. Thereafter, multiple future states can be estimated with given fixed step time T:

$$\begin{aligned} \boldsymbol{x}_1 &= \boldsymbol{A}(T)\boldsymbol{x}_0 + \boldsymbol{B}(T)x_{f,1} \\ \boldsymbol{x}_2 &= \boldsymbol{A}(T)\boldsymbol{x}_1 + \boldsymbol{B}(T)x_{f,2} \\ &... \\ \boldsymbol{x}_N &= \boldsymbol{A}(T)\boldsymbol{x}_{N-1} + \boldsymbol{B}(T)x_{f,N} \end{aligned} \tag{4.8}$$

where N is the number of steps to be optimized ($N \geqslant 1$). In this formulation, foot positions $\boldsymbol{x}_f = [x_{f,1}\ x_{f,2}\ ...\ x_{f,N}]^T$ is the optimization variables. The series of CoM states are affected by the series of foot positions. The cost function is defined based on the series of CoM states. Particularly, we are not quite interested in the positions of CoM after each step, the velocities are more meaningful. Given a reference average velocity $\dot{x}^*$, the cost function could be defined as:

$$\min_{\boldsymbol{x}_f} \sum_{i=1}^{N} ||\dot{x}^* - \dot{x}_i||^2 \tag{4.9}$$

If the problem is formulated as an Linear-Quadratic Regulator (LQR) problem, this term can be written as:

$$\min_{\boldsymbol{x}_f} \sum_{i=1}^{N} (\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*)^T \boldsymbol{Q} (\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*) \tag{4.10}$$

it is not complete without another term that regulates the input variables $\boldsymbol{x}_f$. That is why an anchoring term has been added in previous works to reduce the deviation of planned foot placements:

$$\min_{\boldsymbol{x}_f} \sum_{i=1}^{N} \{(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*)^T \boldsymbol{Q} (\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*) + (\boldsymbol{x}_f - \boldsymbol{x}_f^*)^T \boldsymbol{R} (\boldsymbol{x}_f - \boldsymbol{x}_f^*)\} \tag{4.11}$$

Actually better results can be obtained by introducing another term that minimizes the step length which always make sense since the robot could not take infinite large foot steps:

$$\min_{\boldsymbol{x}_f} \sum_{i=1}^{N} \{(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*)^T \boldsymbol{Q} (\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}^*) + \Delta \boldsymbol{x}_f^T \boldsymbol{R} \Delta \boldsymbol{x}_f\} \tag{4.12}$$

where $\Delta \boldsymbol{x}_f$ is the difference vector calculated from $\boldsymbol{x}_f$, more specifically, $\Delta \boldsymbol{x}_{f,i} = \boldsymbol{x}_{f,(i+1)} - \boldsymbol{x}_{f,i} \, (i = 1, ..., N-1)$. From another point of view, minimizing the foot placement difference is equivalent to minimizing average CoM velocity. Closer foot steps result in less fluctuation of CoM velocity for each single support phase which helps to generate more stable walking motion. This is not a surprise based on the theoretical analysis of Predictive Control scheme proposed in [122] which shows that minimizing any derivative of the motion of the CoM of the robot which enforcing the constraints on the position of the CoP results in stable online walking motion.

At this point, the cost function on foot placements has been defined but not the constraints on them. Due to the foot step length minimization term, self-collision between feet can happen if zero desired velocity is given in lateral direction. Constraints

on the foot placements are necessary to avoid the self-collision. With the presence of constraints on optimized variables, Quadratic Programming (QP) formulation is more suitable for solving the problem. A simple box constraint relative to the current stance foot is used to prevent the swing foot colliding with the stance foot on inner side and over-stretching on the outer side. For example, to constraint the first step $[x_{f,1}, y_{f,1}]$, the bounding box is defined with respect the stance foot position $[x_{f,0}, y_{f,0}]$. In x direction (forward direction), $[x_{f,0} + dx, x_{f,0} - dx]$ defines the limits of the box on both sides of the stance foot. In y direction (lateral direction), $[y_{f,0} \pm dy_{min}, y_{f,0} \pm dy_{max}]$ defines the limits of the box on the side away from the stance leg (the sign depends on the stance leg: "+" for right support and "−" for left support).

4.2 Versatile Walking Modes

With the new formulation, the robot is able to achieve robust walking in many different scenarios. Robust walking here means that the system can achieve robust performance in the presence of sensor noises, external disturbances, modeling error of the robot and as well as the uncertainties of the surroundings. Here, we focus more on the external part of the uncertainties.

4.2.1 Velocity Guided Walking

When there is no external force disturbance presented, the only command given to the robot is a velocity reference which specifies the desired walking speed in sagittal

and lateral direction. A simple PD controller has been used to regulate the actual walking velocity. Foot steps are automatically generated from velocity command. Then trajectories for feet and CoM are generated based on optimized foot steps and sent to the whole-body controller. Whole-body controller finds out torques for all joints based on given trajectories and meanwhile respect to all kinds of dynamic constraints.

4.2.2 Force Guided Walking

When the robot been pushed, the states changes subject to external forces. Those external disturbances are not desirable in general since they tend to push the robot away from its planed trajectories. However, it is possible to make use of small long-lasting external forces to help achieving guided walking motion. In this case the robot is initially following a reference velocity and the walking pattern is generated accordingly. A small external force could be applied to the robot starting from any moment during the walking. This external force will change the state of robot constantly. Normally, the CoM will be biased towards the force direction. As a result, the MPC controller will correct this bias on the go. In the end, the robot will move in the force direction due to the push. Figure 4.1 shows a typical guided walking scenario in which several pushes have been applied to the robot consecutively in opposite directions. The reference velocity given to the robot is 0.5 m/s in x direction and 0 m/s in y direction. Due to lateral pushes, the robot moves back and force in lateral direction. Figure 4.2 shows corresponding detailed data in $x-y$ plane including CoM trajectories, foot placements and push forces. It can be seen from the plot that the CoM trajectoy does not fluctuate

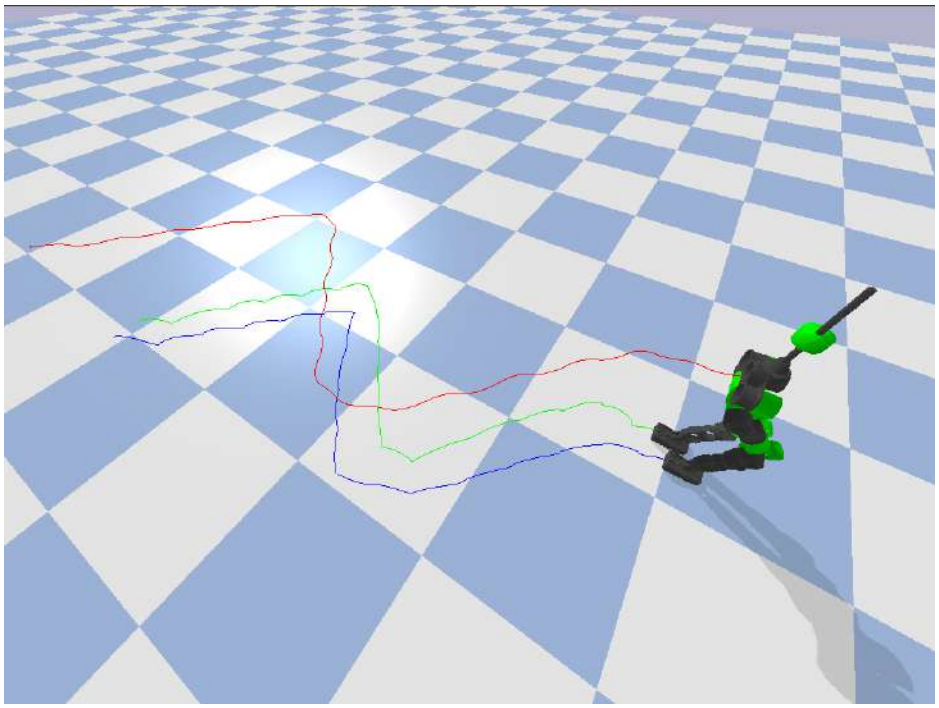


Fig. 4.1 Guided walking scenario. Red trail stands for CoM trajectory, green trail is left foot trajectoy and blue is for the right foot.

that much and the walking pattern is quit dynamic. This is partly because of that a short single support duration has been chosen, here $0.3s$ in this simulation. The other reason is our proposed foot step length minimization term. It is obvious that the foot steps tend to stretch more under external pushes to try to full fill the velocity requirement. Without external push, the foot steps becomes closer to each other.

4.2.3 External Disturbances

For those large forces with short duration, the robot will just take it as external disturbance and behave differently comparing to previous case. In this case, the pushes is different from the previous one in terms of magnitude and lasting duration. Here, large forces with short duration have been applied to the robot as external disturbances.

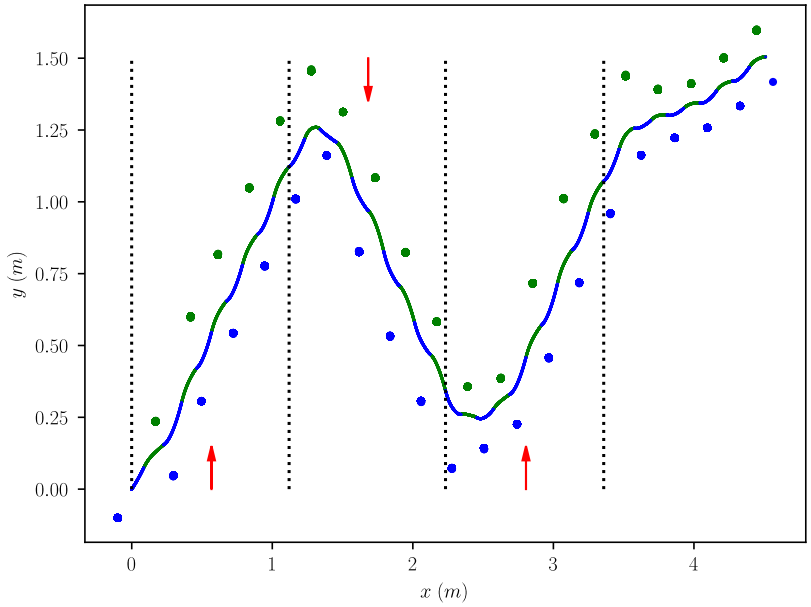


Fig. 4.2 Guided walking data. The velocity reference given the robot is 0.5 m/s in x direction. Three pushes of 20 N lasting for 3 s has been consecutively applied to the waist link of the robot. The first and third one have been applied in positive y direction and the second one in negative y direction as noted with red arrows in the plot. Green dots in the plot stands for left support foot position and blue dots for the right. CoM trajectories also been labeled with different color corresponding to the support phase.

Such kind of disturbance will produce a larger but limited deviation for the robot states. The robot will react to the deviation by taking steps. The velocity reference given the robot is still 0.5 m/s in x direction. Assume that we take the push of $200N$ which lasts for $0.2s$ and apply them to the robot starting from different moments in time which are far away enough from each other so that they do not accumulate. The first and third one have been applied in positive y direction and the second one in negative y direction. Figure 4.3 shows the whole simulation process. The trails of CoM and both feet has been plotted. Detailed data plot is given in Figure 4.4. This case is

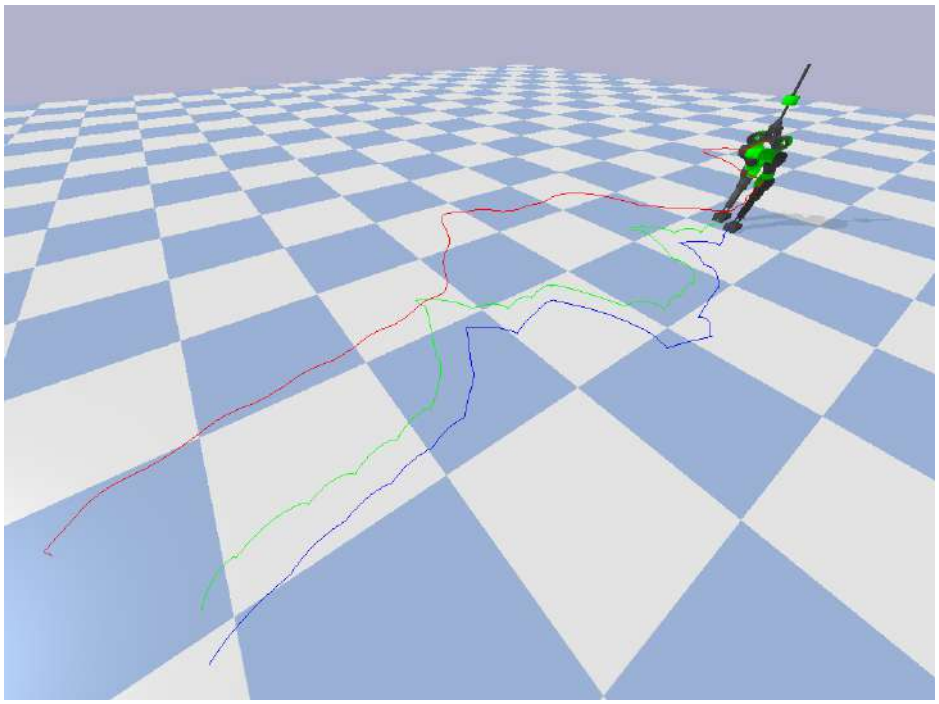


Fig. 4.3 External disturbance scenario. Red trail stands for CoM trajectory, green trail is left foot trajectoy and blue is for the right foot.

more tough for the robot than the previous one. After been pushed, the robot takes several steps to recovery. Huge step length could be spotted in the plot, the largest one is nearly $0.75m$ (the first step right after the second push). What is worse, the stance foot after the step even starts to slid on the ground. This can be known from the non-dot trails of support foot left on the ground. However, the robot is still able to recovery from these pushes and it proves the robustness of the controller.

4.2.4 Environment Uncertainties

Another external uncertainties for a humanoid robot is the environment it interacts with. The environment uncertainties here are those unperceived ones. Without knowing in advance, they can not be modeled and considered in the control system and it can

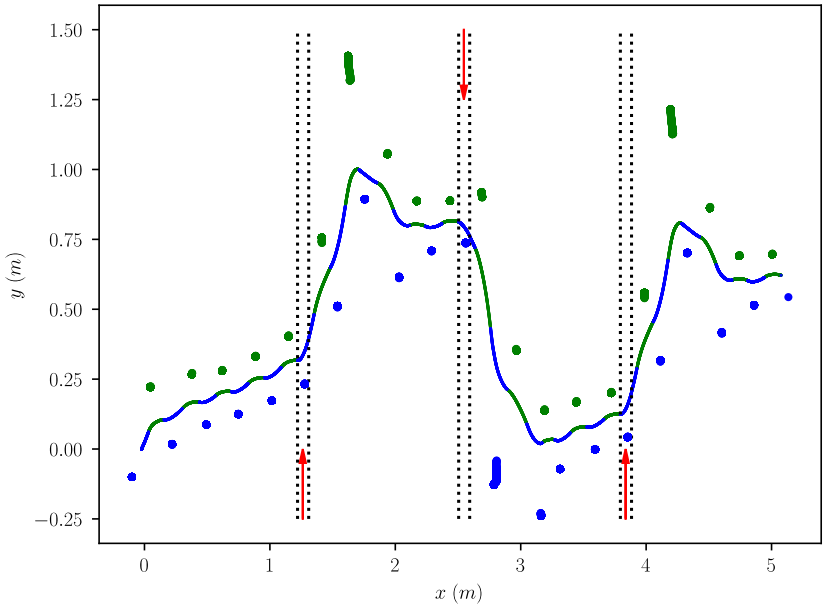
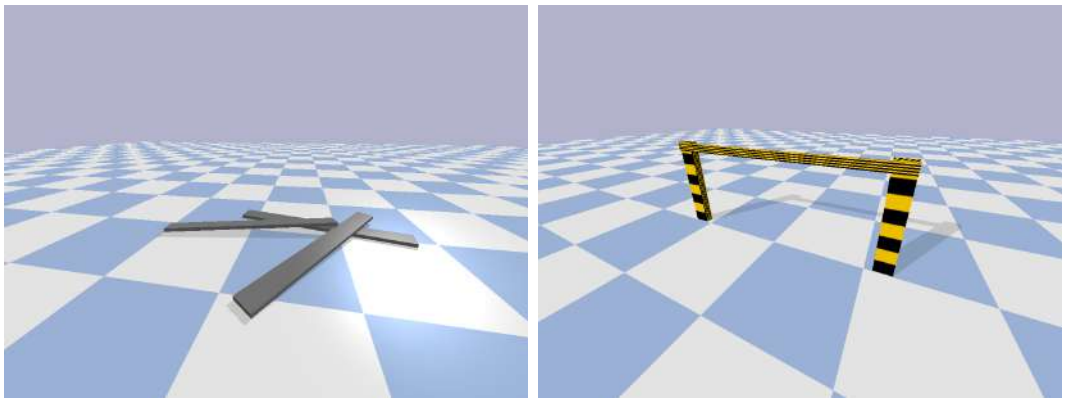


Fig. 4.4 External disturbance data. Three pushes of 200 N lasting for 0.2 s has been separately applied to the waist link of the robot as noted with red arrows in the plot. Green dots in the plot stands for left support foot position and blue dots for the right. CoM trajectories also been labeled with different color corresponding to the support phase.

only be overcome-ed by the robustness or the adaptive capability of the control system. In this part we are going to test two types of unknown environments, one is typical uneven terrain and the other one is unknown barrier as shown in Figure 4.5. For the unknown obstacle scenario, three long boards have been stack together to form the obstacle. Each board has the dimension of $1m \times 0.1m \times 0.02m$ which corresponding to length, width and height respectively. For the unknown barrier, the height is around $1m$ which is higher than the CoM of the robot during walking but lower enough to block upper part of the waist link. In both cases, the robot has neither information about the location nor the dimension of the setups. The only command given to the



(a) Unknown obstacles. (b) Unknown barrier.

Fig. 4.5 Two types of unknown environments.

robot is still a top-level reference velocity which basically command it to walk forward. The robot will perform totally blind walking motion.

Uneven Obstacles

The robot has been command to walk forward with $0.5m/s$ speed. As shown in Figure 4.6, the green trail shows the trajectory of left foot and the blue one stands for the for the right foot. Before stepping on obstacles, the foot steps are distributed rather equally. After stepping on obstacles, foot steps has been updated according to actual robot state.

Unknown Barrier

In this case, a barrier has been placed in front of the robot. Without knowing its existence, the robot will crash onto the barrier. The barrier will stop the robot from moving forward however the robot will still try to follow the reference velocity. As a result, the robot can only step in place. In this case, our proposed formulation which

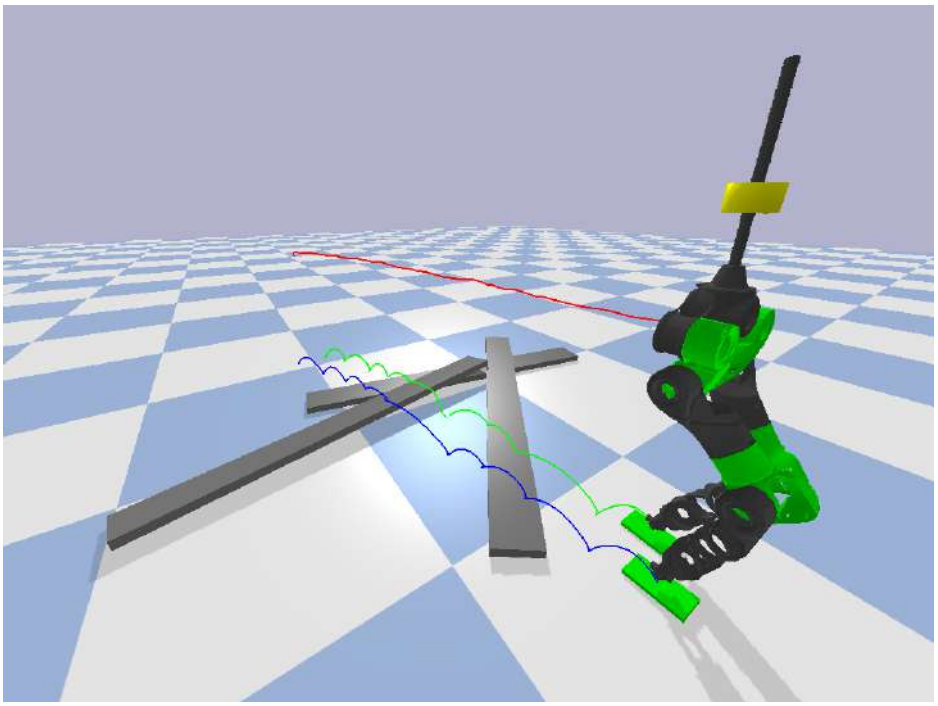


Fig. 4.6 The robot walks over unknown obstacles on the ground.

does not require reference foot step anchoring is critical for the success of stepping in place without falling over. The tails as plotted in Figure 4.7 clearly shows that the robot is stepping in place after being blocked.

4.2.5 Under-actuated Line Feet and Point Feet

Line Feet

The fundamental difference between a planar feet and a line feet is degree of actuation during single support, or the degree of under-actuation of the system. With a planar feet attached to the ankle joint, it could provide full 6D actuation force $\boldsymbol{f}_a = [n_x, n_y, n_z, f_x, f_y, f_z]^T$ but of course with respect to the ZMP and unilateral constraints. With a line feet, the torque around the feet longitude axis can no longer

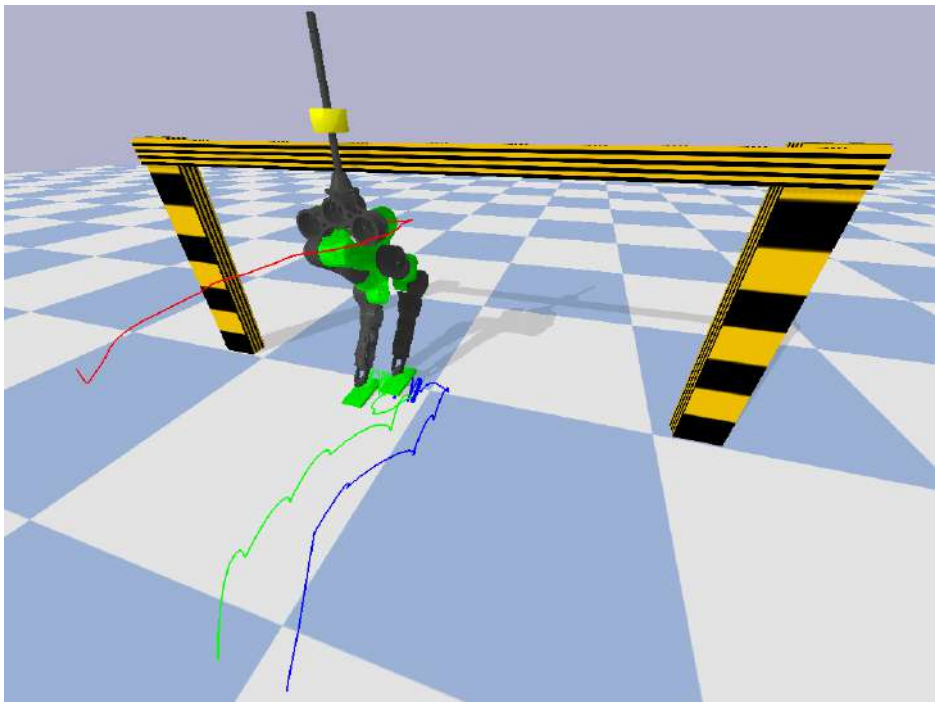
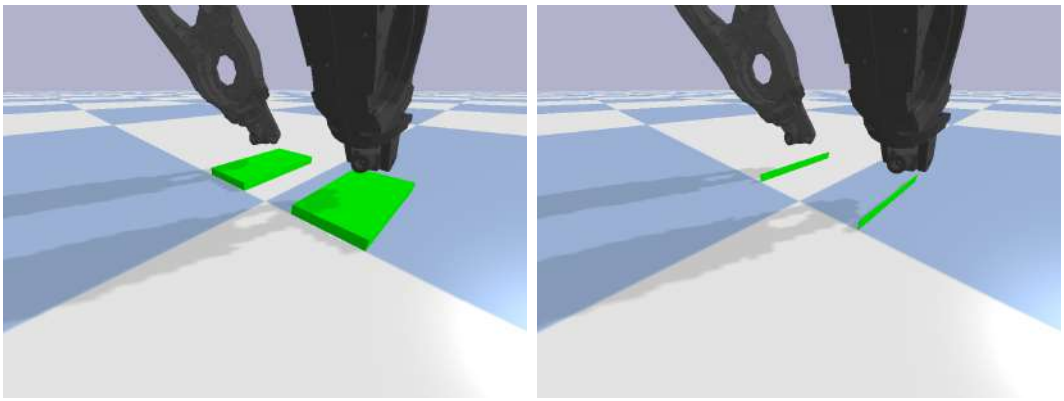


Fig. 4.7 The robot has been blocked by barrier and steps in place.

be provided. With a point feet, all torques around the contact point vanishes which means that the robot can only use contact forces to regulate its motion. However in the planning phase, the LIP template model considered is fundamentally a under-actuated point feet model. It means that in the planning phase, we are indeed considering a under-actuated point feet model. It should be able to generalized to under-actuated robot with point feet. But in practice, even considering a point feet model in control system, a real line feet is desirable since it helps to remove the yaw rotation during walking with the provided friction torque around vertical direction.

To test the feasibility of existing walking controller, we decided to only reduce the size of the feet without modifying the controller formulation. The comparison between feet with different sizes are shown in Figure 4.8. The two pair of feet only differ in



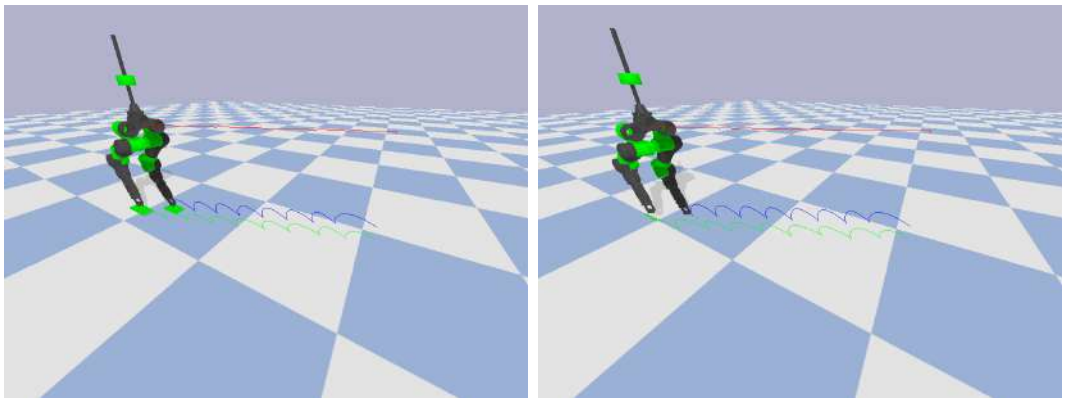
(a) Planar feet. (b) Line feet.

Fig. 4.8 Feet with different sizes. All feet has the same length of $0.2m$ and height of $0.015m$, but their width differ from each other. For planar feet, the width is $0.1m$. For the line feet, the numbers is $0.001m$.

width. The line feet width we use here is $0.001m$ which should be able to be considered as line feet since it provides nearly zero torque around the feet longitude direction. Based on the different feet sizes, the ZMP constraints as defined in (3.7) in whole-body controller will be updated accordingly.

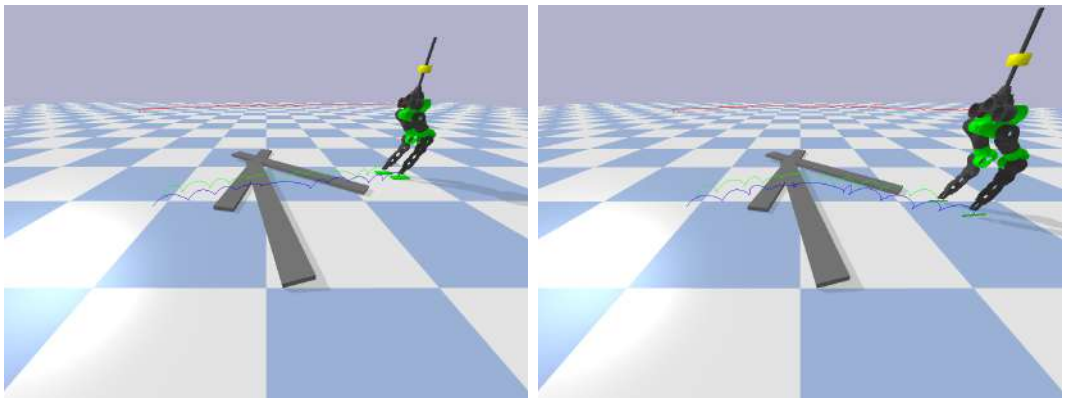
Here, we first compare basic walking motion on flat ground to check the performance of the controller. The results are plotted in Figure 4.9. The controller can handle the line feet very well and the performances are consistent only with minor difference at the end point position.

Next we are going to repeat the same walking tests with unknown obstacles. Results are shown in Figure 4.10. Walking over obstacles enlarge the performance differences. From the trails, one can tell the foot steps are quit different. For planar feet, the robot finally goes over all third board. However for the line feet, the robot has been deviated to the right after stepping over the first two boards and never steps on the



(a) Planar feet. (b) Line feet.

Fig. 4.9 Walking simulation with different sized feet. In both cases, the robot walked for a period of 5 seconds with the same reference velocity.



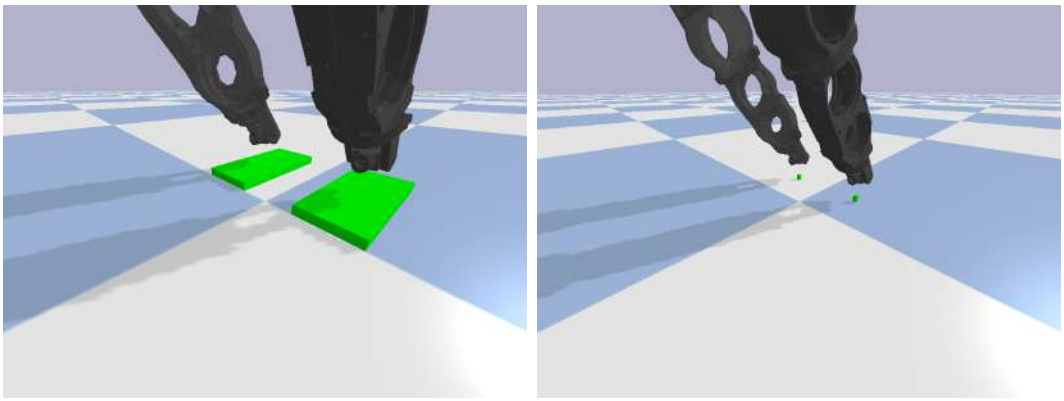
(a) Planar feet. (b) Line feet.

Fig. 4.10 Walking over obstacles with different sized feet. In both cases, the robot walked for a period of 7 seconds with the same reference velocity.

third one. This is largely due to the reactive properties of the control system. But still, the controller is very robust and performs well.

Point Feet

With line feet, the robot is still able to achieve static standing balance. But with two point feet, the robot has to move all the time to keep its balance. If the robot stops stepping, it will definitely fall. Point feet mathematically should be a point without



(a) Planar feet. (b) Point feet.

Fig. 4.11 Comparison between planar feet and point feet. The size of the planar feet is $0.2m \times 0.1m \times 0.015m$. The size of the point feet is $0.01m \times 0.01m \times 0.015m$.

sized dimension. Practically, people use small spheres to approximate the point feet. If the spheres are rigid enough, the contact between it and the ground would be a point with minor position shift. In our case, the easiest way to approximation a point feet is to modify the size of the existing feet in both longitudinal and latitudinal directions. The modified feet is shown in Figure 4.11. The length and width of the feet is both $0.01m$. The feet is small enough to be considered as point feet.

With this point feet, we are going to repeat the walking task first. In this case, there is no preparation phase, meaning that the walking controller starts to engage right after the robot has been spawned in the simulator. The results is shown in Figure.

To test the robustness, the second scenario will be the unknown obstacles. With a very small feet, it becomes very difficult for the robot to establish firm contact (contact with more than two contact points) between its feet and the obstacles. Especially when the obstacle is placed inclined in the world. When the robot steps on it, with small disturbances, the foot tilt and starts to slid. As it slides away, the situation

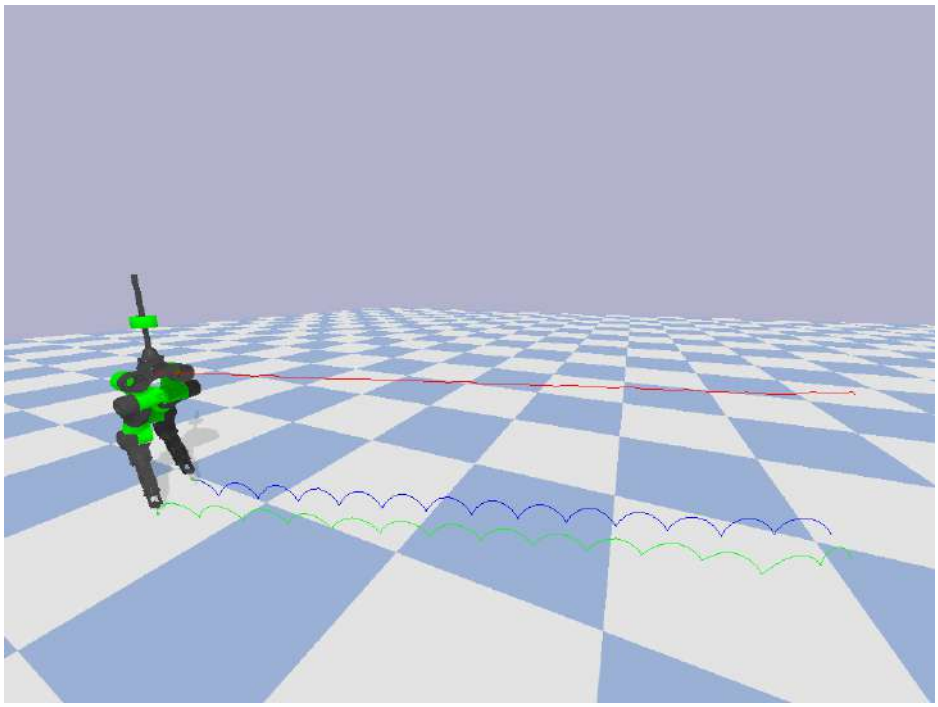


Fig. 4.12 Robot walking in simulation with point feet.

becomes worse and worse and finally leads to an unavoidable fall. For this reason, we replaced the second board a little bit to avoid very inclined surface for the robot to step on. However this is not a problem for a real robot, with a rubber point feet, the robot would be able to walk on inclined surface without any sliding. The simulation is plotted in Figure 4.13. The robot is able to walk robustly with just point feet due to the reactive capability of the MPC controller.

The last test is the barrier blocking test. Without further tuning of parameters, it works similar to the planar feet case which again proves the robustness of the controller.

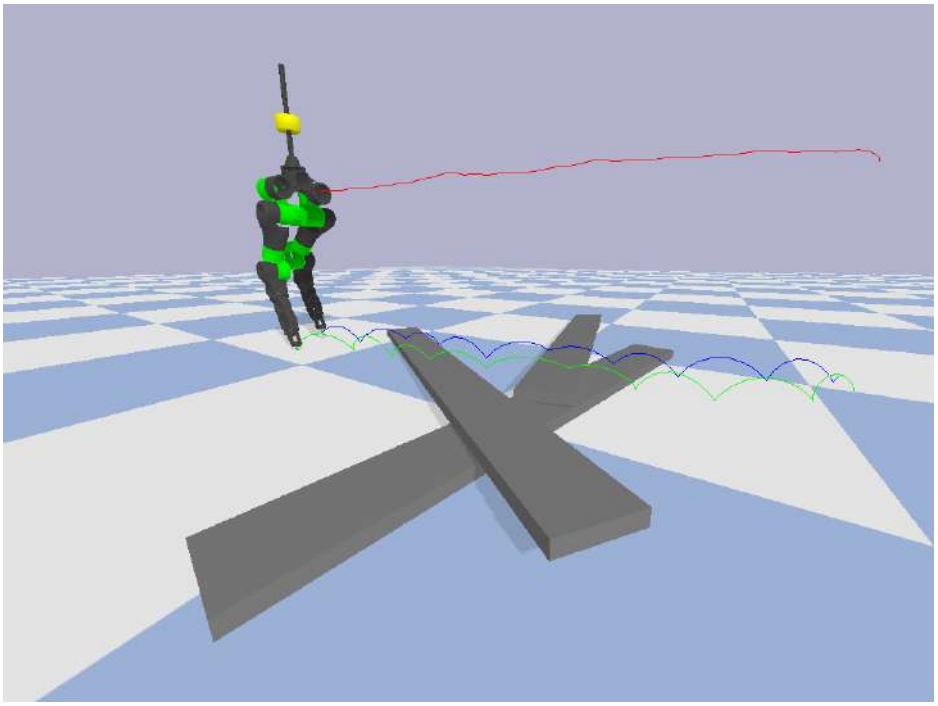


Fig. 4.13 Robot walking in simulation with point feet.

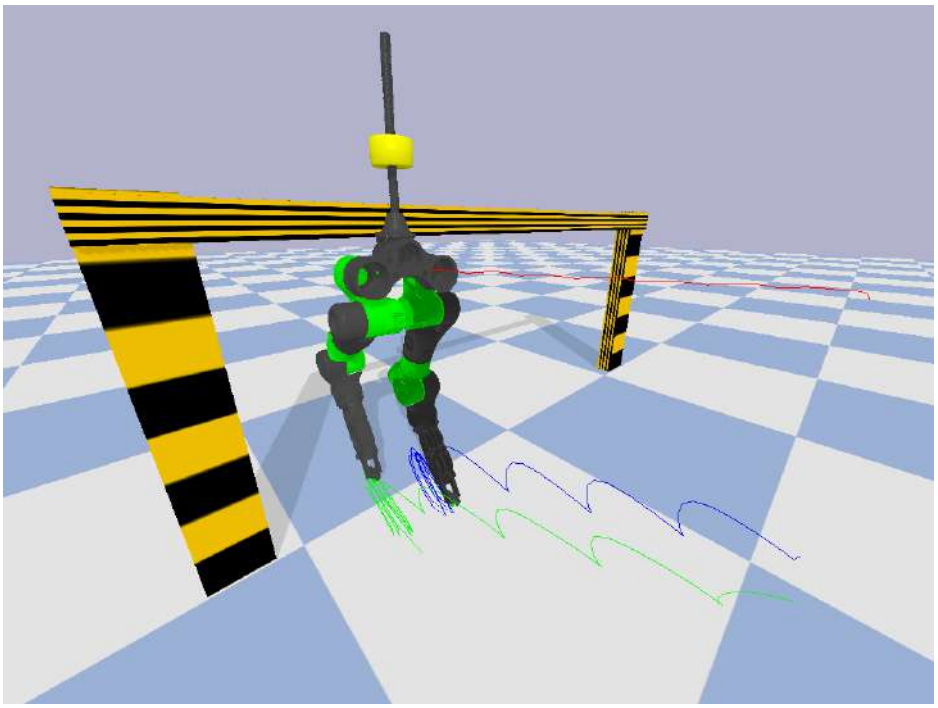


Fig. 4.14 Point feet robot blocked by barrier and steps in place.

4.2.6 CoM Height Updating

Previously it has been shown that the robot can handle uneven terrain by fast re-planning of the MPC scheme. There is a state machine behind strictly counting the time and command the controller to switch between different support phases. With this implementation, the unevenness of the terrain is mostly handled by the whole-body torque controller but not the planner. It is possible to consider the unevenness during the planning phase by updating CoM height based on current landing point position. We still keep the time-based state machine, the robot is commanded to switch support phase based on the output command from state machine. Terrain unevenness would result in incorrect feet height, keeping the CoM still on the same level would lead to considerable amount of mismatching between simple-model planning and simulated full-body robot. Updating CoM height with respect to current support foot would help to reduce the mismatch. At the switching moment, new CoM height can be calculated from current CoM position and future support foot position (the one would be used as support foot during the ensuing support phase right after the switching moment). With this updating, the robot is re-planning based on local support foot frame in each support phase. This helps reduce error introduced by terrain unevenness. Comparing between enabling and disabling this CoM updating has been plotted in Figure 4.15.

4.2.7 Straight Knee Walking

In all of our walking simulation, the robot utilize bent knees. Bent knee helps to avoid singularity and increases the stability of the robot but at the cost of resulting highly

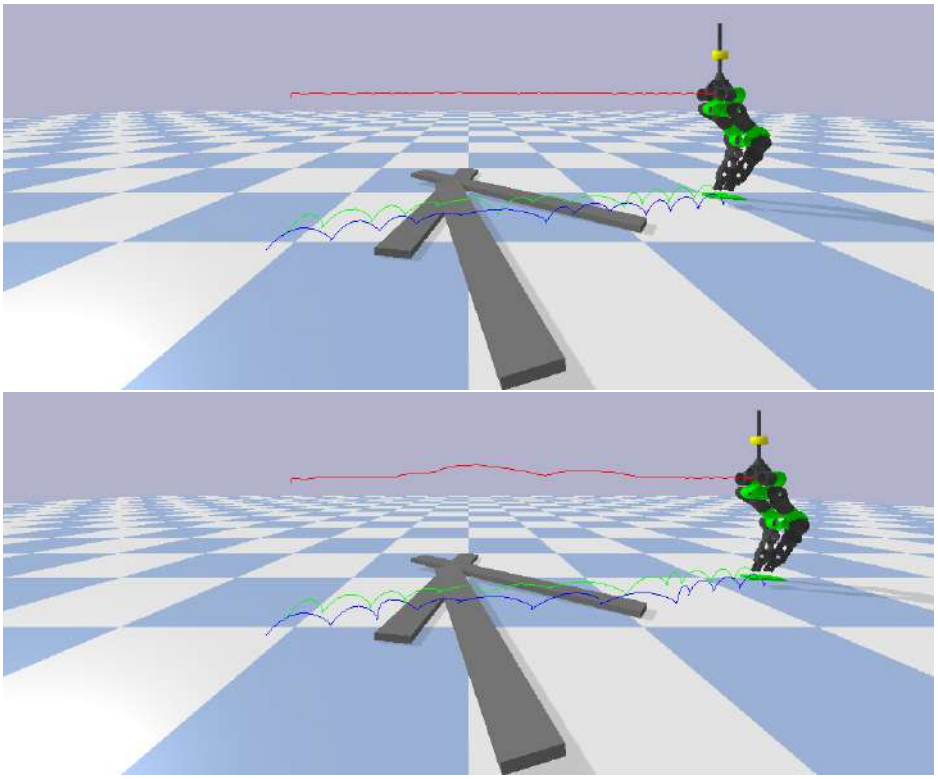


Fig. 4.15 CoM height updating during planning. The above one is without CoM updating and the robot keeps constant CoM height during the whole walking process. Below is the case with CoM height updating enabled.

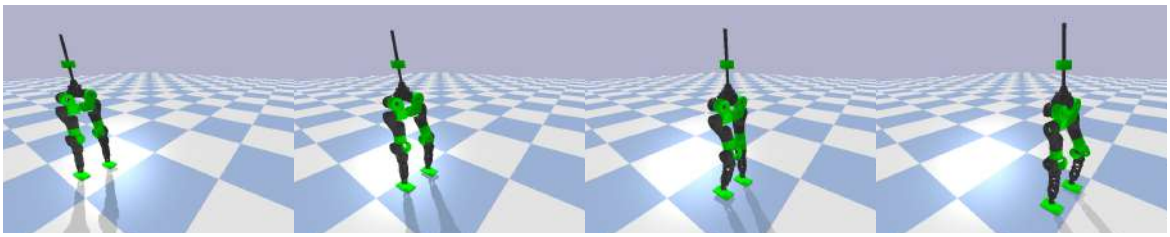


Fig. 4.16 Straight knee walking Simulation.

unnatural and power consumption gaits. On the other spectrum, pure passive walker can achieve highly efficient walking with straight legs. Previously, we addressed this issue by setting a desired CoM height slightly over the robot's reachable height to encourage the robot straighten its knees while all the other kinematic and dynamic constraints are considered in the whole-body controller at run time [123]. By allowing height variations in whole-body controller, there exists a mismatch between the planned motion and resulted whole-body motion even it has been proved this error is limited and can be covered by the ZMP allowance. The simulation results in Figure 4.16 shows that it is very effective in generating straight knee walking motion. The idea could be illustrated by the Figure 4.17. Setting the CoM reference higher than the reachable height is equivalent to adding a spring damper between the reference and the actual one. The virtual spring will pull the CoM upward and this idea is actually similar to the virtual model control [50] in spirit.

Above mentioned method tries to resolve the straight knee problem in the whole-body controller level, anther option is to solve it in the planning level. With a vertical motion of the CoM decided before-hand (for example an arc curve), the height variance could be compensated by iterating the linear MPC scheme. At each MPC re-planning

initiating moment, the current CoM height will be used to update LIPM accordingly, the updated model will be used to plan the motion for current iteration. In such a manner, the CoM height variation could be incorporated into the existing MPC scheme.

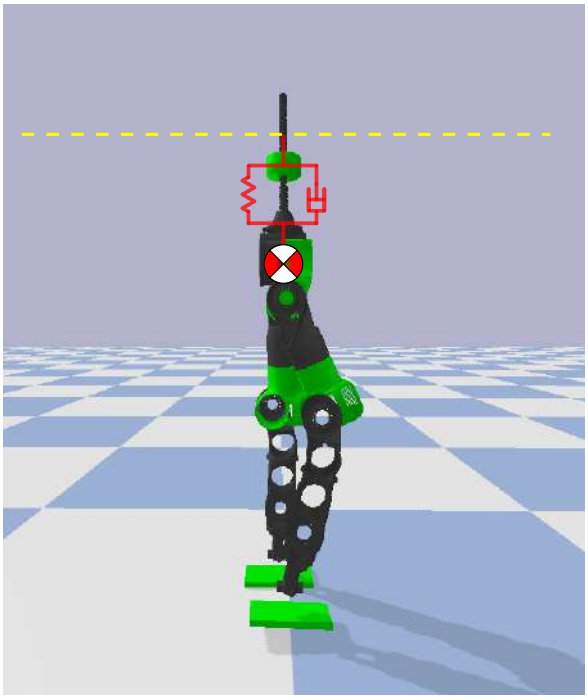


Fig. 4.17 Setting the CoM reference (yellow dash line) higher than the reachable height is equivalent to adding a spring damper between the reference and the actual one. In the figure, the reference has been plotted much higher than the reachable one just to illustrate the idea.

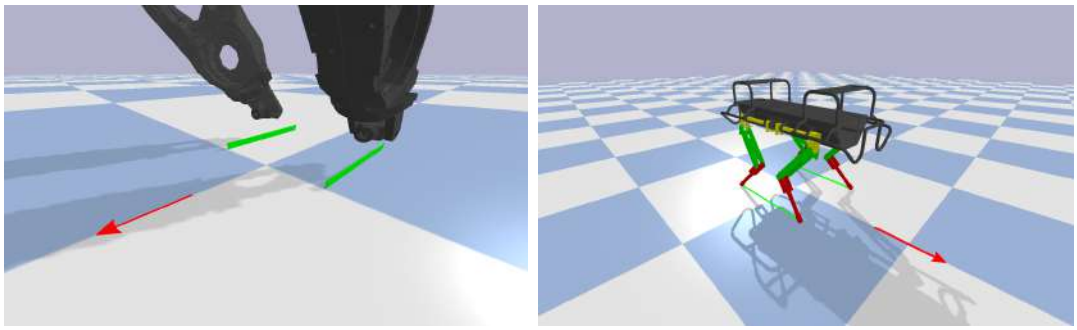
4.2.8 Event Based Re-planning

About described CoM height updating scheme is based on time state machine so it could be called time-based CoM height updating. It improves the robot performance comparing to the constant height case. However it still suffers from some minor issues. At the moment to switch support, the swing foot reference could be either already

penetrating into the ground (early landing) or still in the air (late landing). For the early landing case, the penetrating reference below contact surface would lead to significant contact force increase. For the late landing case, the current foot position in the air has been used to update CoM height is definitely higher than the one when the foot touch the ground during the ensuing support phase. In both cases, time based state machine leads to undesired behaviors. A solution is to use swing leg touchdown event to trigger the re-planning instead of waiting for command from the time driven state machine. The event triggered re-planning only happens occasionally when swing foot touch the ground, during the step, the re-planning of MPC is still time driven assuming constant step time. In the end, it is a time-event hybrid driven re-planning scheme.

4.2.9 Extension to Quadruped

In [124], Ivan Sutherland claims that multi-legged robot gait can be mapped into virtual biped one-foot gaits if all support legs are coordinated to act like a virtual leg. In [125], Marc Raibert points out that three typical types of dynamic quadruped gaits such as trot, pace and bound can be mapped to biped walking gait by using legs in pairs. Diagonal pairs of legs are used for trot gait, lateral pairs for pace gait and front/rear pairs for bound gait. Using legs in pairs can transform quadruped gaits into a common underlying gait, virtual biped gait. Thus the algorithms developed before for biped robots can be adopted to produce all three quadruped gaits. Compassion between humanoid walking motion and quadruped gaits are given in Figure 4.18, 4.19

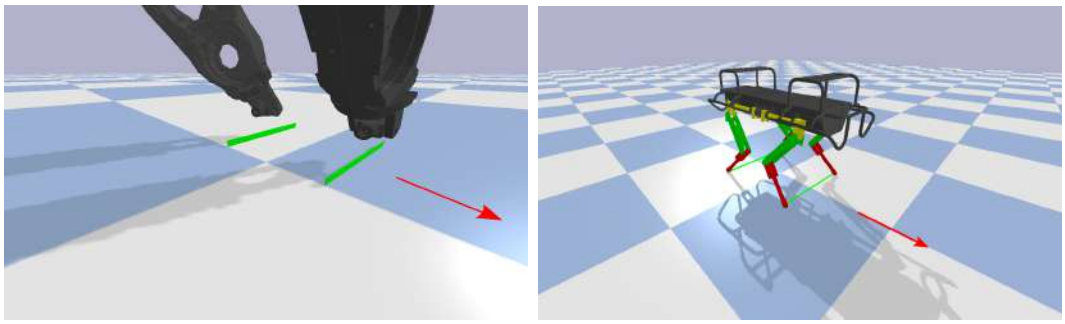


(a) Humanoid with line feet. (b) Quadruped pace gait.

Fig. 4.18 Humanoid robot walking forward with line feet and quadruped pace gait. Lateral pairs of legs resemble the line feet. The red arrow indicates the walking directions.

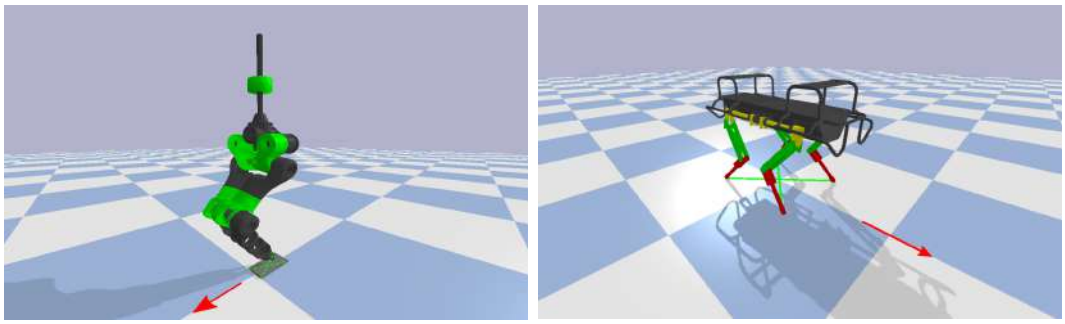
and 4.20. For the quadruped trot gait as shown in Figure 4.20, there is no exact biped gait mapping to it. A humanoid robot with planar feet overlapping on top of each other has been used to compare. Practically, the previous case is impossible to realize due to self collision between feet. It is just for illustrating the analogy between this two cases. For the quadruped, it does not have the self-collision problem and this is actually beneficial for the robot. Due to the self-collision between feet, the humanoid robot has to separate its feet and this leads to unavoidable swing motion in lateral direction. Without this intrinsic limitation, the quadruped robot could achieve zero swing motion in theory. Meaning that the virtual leg is exactly controlled under the CoM of the robot. For real hardware, with small deviation of CoM, the error can be easily corrected with small virtual foot steps.

In order to reuse the planner and controller developed for biped robot, some modifications are needed to generalize the framework for both biped and quadruped. Here only some key modifications are explained.



(a) Humanoid with line feet. (b) Quadruped bound gait.

Fig. 4.19 Humanoid robot walking sideways and quadruped bound gait. Front/rear pairs of legs resemble the line feet when the humanoid robot walking sideways. The red arrow indicates the walking directions.



(a) Humanoid with planar feet overlap. (b) Quadruped trot gait.

Fig. 4.20 Humanoid robot walking forward with overlapped feet and quadruped bound gait. Diagonal pairs of legs resemble the planar feet of a humanoid robot when the feet of the robot stack together.

Whole-body Controller Generalization

For the whole-body controller, in order to deal with biped robot and quadruped robot, the GRF needs to be generalized. For biped robot, planar feet are normally equipped. The GRF acting on each foot could be represented by a spatial force $\boldsymbol{f}_s^i = [n_x^i, n_y^i, n_z^i, f_x^i, f_y^i, f_z^i]^T$ which consists of a pure couple $\boldsymbol{n}^i = [n_x^i, n_y^i, n_z^i]^T$ and a pure force $\boldsymbol{f}^i = [f_x^i, f_y^i, f_z^i]^T$. However for a quadruped robot with point feet, this representation is redundant since it is basically under-actuated and can only get pure force from the contact point but not couple. The same applies to biped robot with line feet or point feet. This motivates me to choose another representation for GRF. It has been explored in several previous works [126] [59]. The basic idea is to represent the spatial force $\boldsymbol{f}_s^i = [n_x^i, n_y^i, n_z^i, f_x^i, f_y^i, f_z^i]^T$ as a combination of pure forces $\boldsymbol{f}^{ij} = [f_x^{ij}, f_y^{ij}, f_z^{ij}]^T$ (i: support foot index, j: vertex index) acting at the contact polygon vertices. As shown in Figure 4.21, various types of contacts can be represented in a unified form. For each vertex force $\boldsymbol{f}^{ij}$, friction constraints (3.5) and unilateral constraints (3.6) can be easily enforced. What is beneficial of this formulation is that the (3.7) is inexplicit enforced by the unilateral constraints (3.6). That is to say, by constraining the z-component for all vertex forces to be positive means that the foot has to stick to the ground. With all positive z-component, the average CoP (ZMP) will stay inside the support polygon by definition.

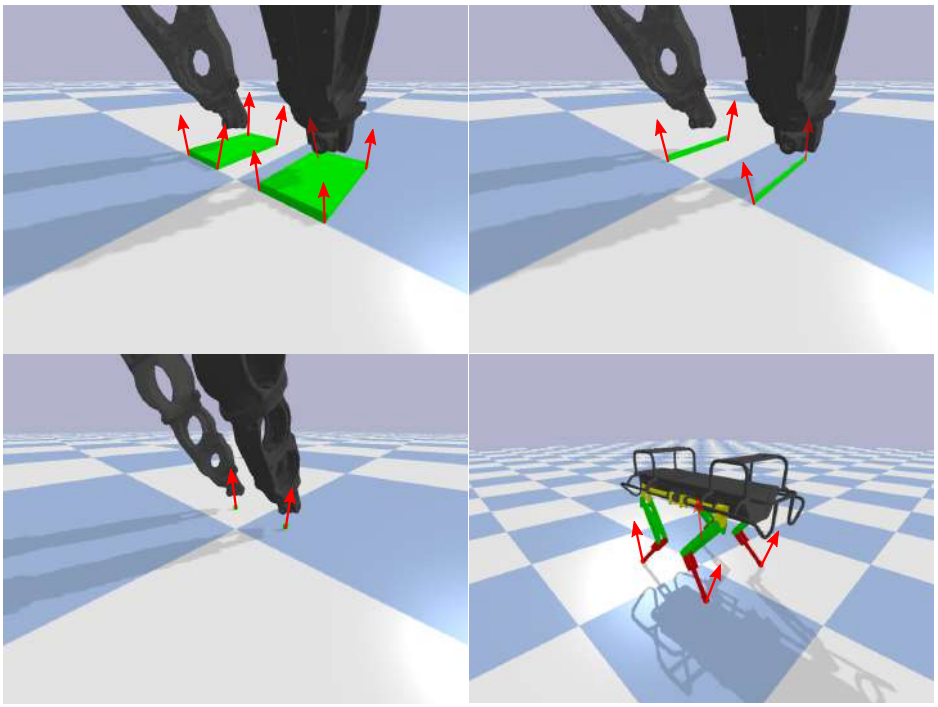


Fig. 4.21 GRF vertices representation.

Walking Motion Planner Generalization

For both biped and quadruped walking, the same LIPM template model will be used, so there is no essential difference between them. The same MPC scheme will be used to produce the walking references as explained in 4.1.2. The cost function (4.12) stays unchanged however the constraints based on whole-body kinematics needs to be adapted. For example, for biped robot, the feet clearance constraints are necessary to avoid self-collision. This no longer holds for quadruped robot. More specifically, this feet clearance constraints various depending on what gait the quadruped robot is using. For trot gait, since the gait planning is based on virtual legs, the self-collision problem will never happen. This constraint vanishes for imaginary trot legs. For pace and bound gait, the situation is slightly different. Physically it is feasible for quadruped

robot to cross the virtual legs as long as the physical legs does not collides with each other. When bounding forward, the font legs could be narrowed to allow the rear legs swing past. When pacing forward, under situation such as being pushed to the right, the left legs could be narrowed or widened to swing over right stance legs to avoid collision.

For quadruped robot, the planner is planning gaits for virtual leg, a mechanism that maps virtual leg behavior to physical legs are needed, and vice versa. At touchdown moment, the average position of swing feet will be calculated and used as virtual stance foot position which is needed for planning. Current CoM states will be also used as initial state for planning. Based on those information, the planner will plan virtual foot placements and CoM trajectories for next few steps (normally two steps are enough). The very first planned foot placement will be used to generate swing trajectory for the virtual leg. The planner only gives the foot placement for the virtual leg, the foot placement needs to be mapped to physical leg pair. The foot placement for the virtual leg can be thought as the averaged position of the physical leg pair, but it still leaves two degrees of freedom unspecified: the distance between the physical feet that form the virtual leg and the yaw orientation of the line connecting the feet [125]. The distance could use a predefined constant value and the yaw orientation of the line connecting the feet could use the orientation of the base link. In such a way, the robot is able to walk in x-y direction with fixed yaw orientation. In fact, the yaw control can be incorporated into this leg coordination mechanism. As shown in Figure 4.22, trot gait assumed, it shows the touchdown moment, LF/RH legs are switching from stance

phase to swing phase. The virtual swing foot is calculated from LF/RH leg pair. After the next foot placement has been planed, yaw offset has been considered to calculate the final foot placements for the swing leg pair.

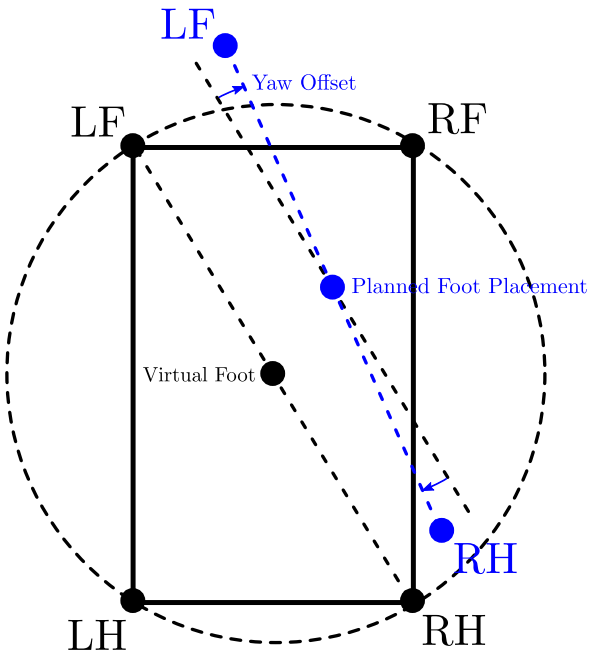


Fig. 4.22 Foot placement with yaw control. Final planed foot placement have been plotted in blue color.

State Machine Generalization

For biped walking motion, no double support phase has been assumed. We will keep similar assumption for all dynamic gaits (trot, pace and bound) for quadruped robot which assumes that no more that two contact points exists during the whole walking motion. That is to say, the legs of the quadruped robot works in pairs and at any time only one pair of legs stay in contact with the ground. For biped walking, the robot switch between left foot and right foot with fixed duration. To reuse the state machine developed for biped walking, we also decided to use the same switching pattern. The

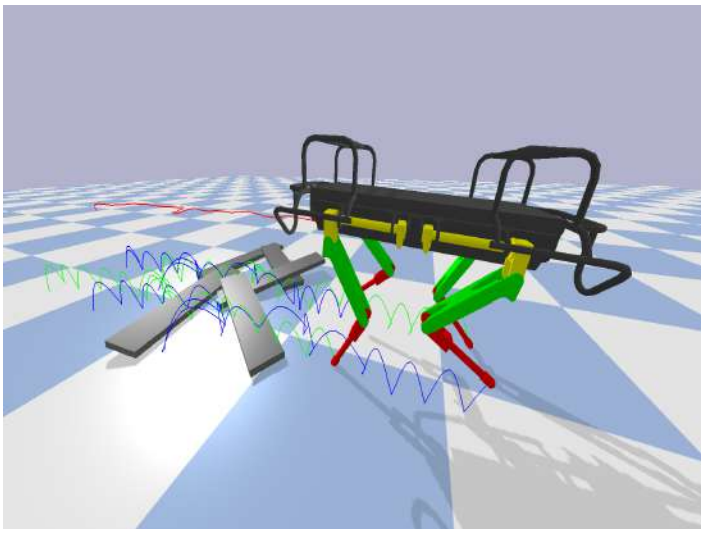
only difference is that the quadruped robot switch between pairs of legs instead of single leg.

Simulation Results

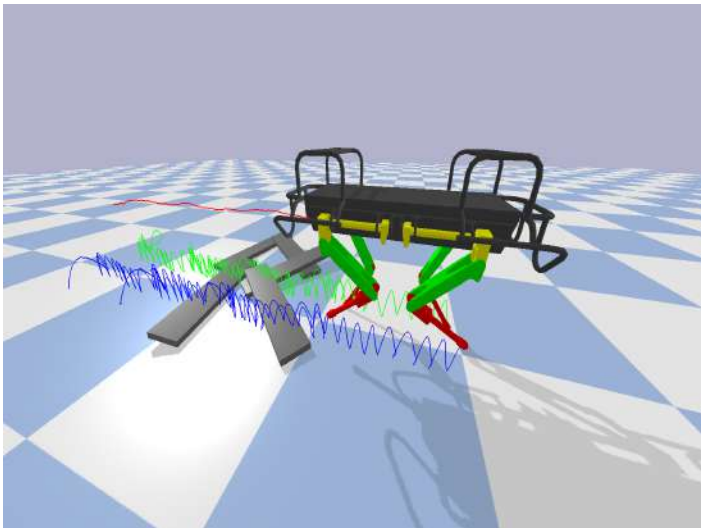
To validate the feasibility of extending the biped control framework to quadruped, several walking simulation have been conducted on HyQ [127], a hydraulically actuated torque-controllable quadruped robot. In order to test the robustness, several obstacles have been added into the scene. The top-level command given to the robot is a forward velocity and all the rest gaits and whole body motion are generated autonomously. The results is plotted in Figure 4.23. When walking over obstacles, the robot adapts its footsteps due to unexpected footholds variations. Different gaits require different parameters but the overall framework remains the same and it proves its generality.

4.3 Cross-Step

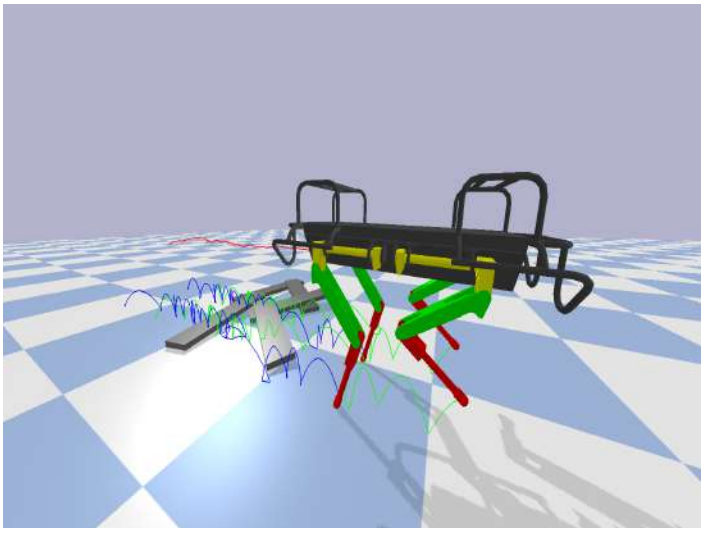
The comparison between humanoid and quadruped inspired me to explore the possibility of cross-step on humanoid robot. For a quadruped, the cross-step motion is more naturally emerged. As shown in Figure 4.24, the BigDog has been disturbed from one side, during the recovery process, its forelegs cross together. This cross-step motion is achievable on a quadruped with staggered feet and it is relatively easier for them due to the fact their legs are long and thin. For a humanoid robot, its legs are usually more bulky and therefore more problematic for this type of motion due to self-collision.



(a) HyQ trots over obstacles. LF/RH pair trajectory in green, RF/LH pair trajectory in blue. CoM trajectories is plotted in red color.



(b) HyQ paces over obstacles. LF/LH pair trajectory in green, RF/RH pair trajectory in blue. CoM trajectories is plotted in red color.



(c) HyQ bounds over obstacles. LF/RF pair trajectory in green, LH/RH pair trajectory in blue. CoM trajectories is plotted in red color.

Fig. 4.23 HyQ walks over obstacles with different gaits.

Fig. 4.24 Boston Dynamics BigDog takes large side step after been heavily disturbed from side. In the last plot (lower left), the two front legs cross together.

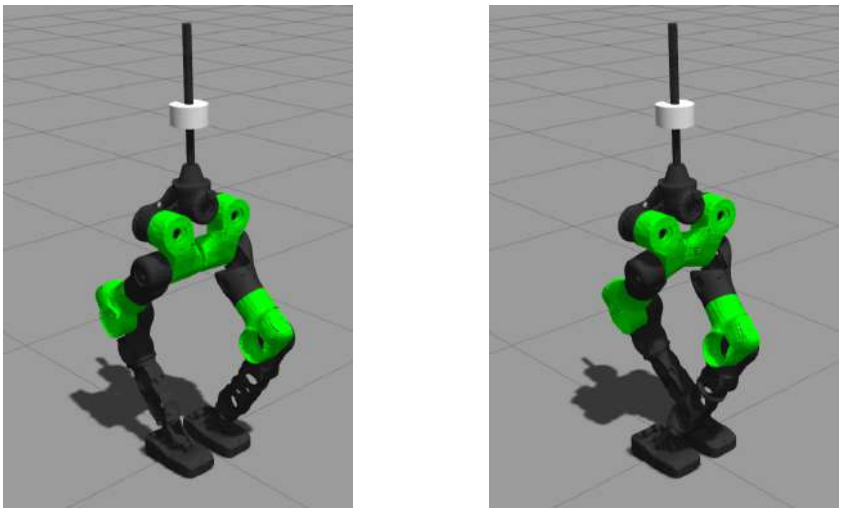
In the next chapter, two new configurations are introduced for humanoid robot which enables the robot to do cross-step motion.

In this part two new configurations for humanoid robot balancing and locomotion have been explored. Centroidal momentum manipulability analysis has been performed to study the features of the newly proposed configurations. Numerical simulations show that they outperform the regular ones in terms of angular momentum manipulability. More than that, the new configurations allow the humanoid robot to perform cross-step motions which is usually risky or mechanically impossible for most existing robots. However, cross-step introduces non-convex feasible region which makes it difficult to be incorporated into the existing step planner as explained in 4.1.1. Therefore, a simple heuristic has been proposed to help choosing a sub-convex region for the step planner. To validate the cross-step movement, walking simulations have been performed.

4.3.1 Introduction

For a humanoid robot, maintaining its balance is usually the first priority to guarantee. Many criteria has been proposed to evaluate its stability, and thus help designing of balance or locomotion controller. The most commonly used dynamic stability criteria is zero moment point (ZMP)[128, 33] or center of pressure (CoP), it is required to stay inside the support polygon for all time. Foot rotation indicator (FRI) [129] requires the foot has no rotation. Zero Rate of change of Angular Momentum (ZRAM) [130] guarantees rotationally stability. Capture point (CP) [37] defines a point on the ground where the robot can step to in order to bring itself to a complete stop. All these criteria summarizes the robot stability on a reduced dimension geometry point and this compression unavoidably cause the loss of information. For example, infinite configurations which are stable could ended up with the same ZMP (or CoP).

Most model-based balancing or locomotion planner use a simplified model to represent the essence of a high degree of freedom multi-rigid-body system. Based on the template model, planner often generates Center of Mass (CoM) and end-effector references for the humanoid robot to track. The ability to closely track those references becomes extremely important for system controllability and stability. Manipulability of end-effector is proposed for measuring this ability and it has been well studied [131–133]. Correspondingly, this manipulability concept has been extended to ZMP point [134] and CoM point [135] [136] [137] [138]. Furthermore, centroidal momentum manipulability concept [71] has been proposed to quantify system linear and angular momentum manipulability.



(a) *forward/backward* (b) *twist*

Fig. 4.25 Two new configurations for humanoid robot which enables it to perform cross-step. The left one is called *forward/backward* configuration: left knee bending forward and right knee bending backward. The right one is called *twist* and can be achieved from the left one by one more crossing legs action.

In this paper, two new configurations for humanoid robot have been proposed first as shown in Figure 4.25. Centroidal momentum manipulability analysis has been performed on these two configurations, as well as two other conventional ones. Based on the newly proposed configurations, cross-step possibility has been explored. In the end, walking simulations have been performed to show the viability of cross-step movement.

4.3.2 Centroidal Momentum Manipulability

The Centroidal Momentum Matrix (CMM) relates the robot's generalized velocities to its centroidal momentum [71]:

$$\boldsymbol{h} = \boldsymbol{A}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{4.13}$$

where $\boldsymbol{h} \in \mathbb{R}^{6\times1}$ is the centroidal momentum, $\boldsymbol{A} \in \mathbb{R}^{6\times(6+n)}$ is the CMM, $\dot{\boldsymbol{q}} \in \mathbb{R}^{(6+n)\times1}$ is the generalized joint velocity which consists of the floating-base velocity $\dot{\boldsymbol{q}}_b = [\boldsymbol{v}_b, \boldsymbol{\omega}_b] \in \mathbb{R}^{6\times1}$ and actuated joint velocity $\dot{\boldsymbol{q}}_a \in \mathbb{R}^{n\times1}$.

Centroidal momentum manipulability and corresponding ellipsoid are also proposed in [71]. Due to the scale disparities between the linear and angular part of the system momentum, it is preferred to construct two ellipsoids separately. More specifically, (4.13) can be expanded:

$$\begin{bmatrix} \boldsymbol{l} \\ \boldsymbol{k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_l \\ \boldsymbol{A}_k \end{bmatrix} \dot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{A}_{lb} & \boldsymbol{A}_{la} \\ \boldsymbol{A}_{kb} & \boldsymbol{A}_{ka} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_b \\ \dot{\boldsymbol{q}}_a \end{bmatrix} \tag{4.14}$$

where $\boldsymbol{l} \in \mathbb{R}^{3\times1}$, $\boldsymbol{k} \in \mathbb{R}^{3\times1}$ are centroidal linear momentum and angular momentum, $\boldsymbol{A}_l \in \mathbb{R}^{3\times(6+n)}$ and $\boldsymbol{A}_k \in \mathbb{R}^{3\times(6+n)}$ are corresponding linear and angular momentum matrix. The subscript b and a indicate the base related part and configuration related part of corresponding momentum matrix. More specifically, $\boldsymbol{A}_{lb} \in \mathbb{R}^{3\times6}$ maps floating base velocity to system linear momentum and $\boldsymbol{A}_{la} \in \mathbb{R}^{3\times n}$ maps the actuated joint velocity part.

Given the matrix $\boldsymbol{A}_l$, the linear momentum manipulability can be calculated:

$$\omega_l = \sqrt{det(\boldsymbol{A}_l \boldsymbol{A}_l^T)} \tag{4.15}$$

where $det(*)$ denotes the determinant operation, the index ω_l measures the ability of transferring generalized joint velocity $\dot{\boldsymbol{q}}$ to system linear momentum $\boldsymbol{l}$. Since $\boldsymbol{A}_l$ is

related to the $\boldsymbol{q}$, this index is also configuration related. This index is just a scaler indicator, more information can be visualized by constructing a ellipsoid from the matrix $\boldsymbol{A}_l$ with singular value decomposition (SVD),

$$\boldsymbol{A}_l = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{4.16}$$

where

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_x & \boldsymbol{u}_y & \boldsymbol{u}_z \end{bmatrix} \tag{4.17}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_x & 0 & 0 & ... \\ 0 & \sigma_y & 0 & ... \\ 0 & 0 & \sigma_z & ... \end{bmatrix} \tag{4.18}$$

$$\boldsymbol{V}^T = \begin{bmatrix} \boldsymbol{v}_1^T & \boldsymbol{v}_2^T & ... & \boldsymbol{v}_m^T \end{bmatrix}^T \tag{4.19}$$

The principle axes of the ellipsoid are $\sigma_x\boldsymbol{u}_x$, $\sigma_y\boldsymbol{u}_y$ and $\sigma_z\boldsymbol{u}_z$. It is worth noting that the manipulability can be also calculated from singular values $\omega_l = \sigma_x\sigma_y\sigma_z$. The same calculation also applies for the angular momentum matrix $\boldsymbol{A}_k$ and all sub-matrices in (4.14).

Manipulability Contribution

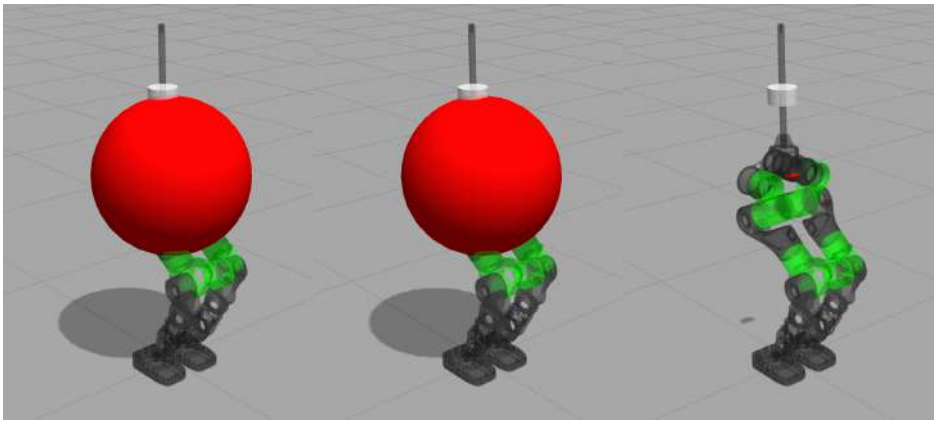
As mentioned before, the system momentum is contributed from floating base velocity $\dot{\boldsymbol{q}}_b$ and actuated joint velocity $\dot{\boldsymbol{q}}_a$. The previous part can be interpreted as base movement related contribution to system momentum, and the later part can be treated

as body movement related contribution. In general, they contribute differently to system momentum. We are going to explore this in simulation with the lower body of our humanoid robot CogIMon.

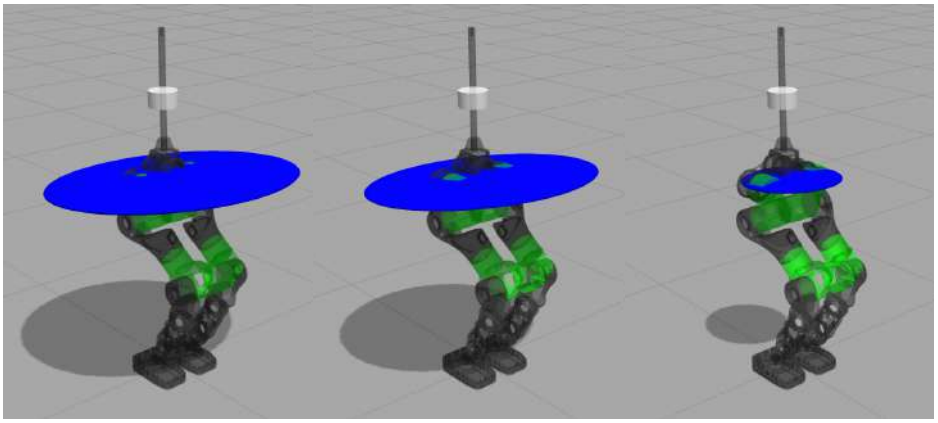
The lower body of CogIMon has 12 actuated DoF (6 for each leg: 3 hip joints, 1 knee joint and 2 ankle joints) [69]. A fake mass link has been fixed to the top of pelvis link to represent the upper body. In simulation, the robot has been command to a given posture (CoM height equals to $0.8m$). Manipulability corresponding to sub-matrices in (4.14) have been computed and listed in Table 4.1. According to the data in the table, the contribution from floating base velocity $\dot{\boldsymbol{q}}_b$ dominant the linear part ($\omega_{la} : \omega_{lb} = 1{:}3375$). However this is not the case for angular momentum, actuated joint velocity $\dot{\boldsymbol{q}}_a$ contributed a comparable part ($\omega_{ka} : \omega_{kb} = 1{:}6$) of angular momentum for the system. It is more straightforward to compare the contribution by observing the different manipulability ellipsoids as shown in Figure 4.26. All the results indicate that the actuated joint velocity $\dot{\boldsymbol{q}}_a$ has very limited contribution to the system linear momentum (or CoM velocity) but has a considerable amount of influence on the angular momentum. As a result, this paper will focus on studying how body movement contributes to the angular momentum of the system.

Table 4.1 Manipulability Contribution

Manipulability	ω	ω_b	ω_a	$\omega_a : \omega_b$
Linear momentum	209575.85	207226.92	61.40	1:3375
Angular momentum	66.03	42.73	6.63	1:6



(a) Linear momentum ellipsoids generated from $\boldsymbol{A}_l$, $\boldsymbol{A}_{lb}$ and $\boldsymbol{A}_{la}$



(b) Angular momentum ellipsoids generated from $\boldsymbol{A}_k$, $\boldsymbol{A}_{kb}$ and $\boldsymbol{A}_{ka}$

Fig. 4.26 Momentum manipulability ellipsoids. For better visualization, a scale factor 10^{-2} has been applied to those linear momentum ellipsoids. Because of the scale disparities between linear and angular momentum, a different scale factors 10^{-1} have been applied to angular momentum ellipsoids. Linear momentum ellipsoids have been plotted in red color and angular momentum ellipsoids in blue for differentiation.

Angular Momentum Manipulability Related To Different Configurations

In this part, four different configurations as shown in Figure 4.27 are going to be examined. The *forward/forward* configuration is just like human with two knees bending forward. It is possible for a robot to bend its knees backward with proper mechanical design and which results in the *backward/backward* configuration. One analogy is the *elbow-up* and *elbow-down* configurations for a manipulator. A mix of the previous two leads to the *forward/backward* configuration. It can be further extended to a *twist* configuration by crossing step the left foot to the right side of the right foot.

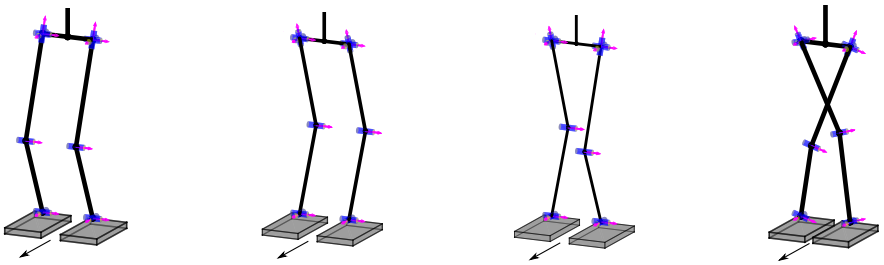


Fig. 4.27 Four configurations of humanoid robot (arrows indicate forward moving direction): (a) *forward/forward*; (b) *backward/backward*; (c) *forward/backward*; (d) *twist*. Here, *forward* and *backward* means knee configuration.

To evaluate the angular momentum manipulability of these four configurations, the feet of the robot are initiated at the same location on the ground (left foot and right foot has been swapped in *twist* configuration) and the CoM is regulated to the same position (x and y take the position of the center of the feet, $z = 0.8m$). The configuration related angular momentum ellipsoids are plotted in Figure 4.28 and corresponding manipulability indexes have been calculated and listed in Table 4.2. It can be seen from

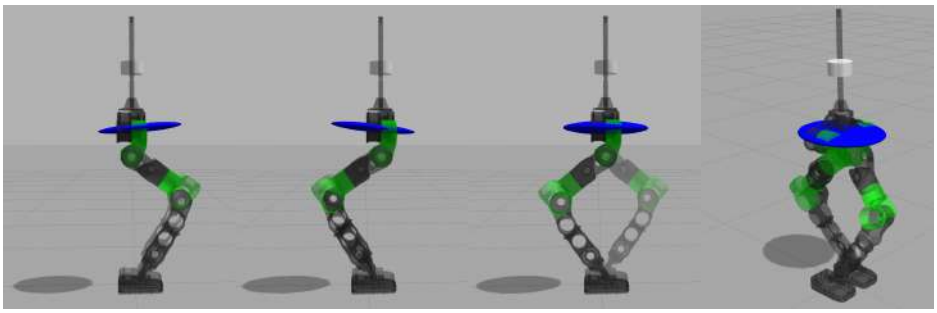


Fig. 4.28 Configuration related angular momentum ellipsoids of four different configurations from left to right: *forward/forward*, *backward/backward*, *forward/backward* and *twist*.

the table that the configuration *forward/forward* and *backward/backward* have similar manipulability. However, these two configurations show different directional features (the first two plots in Figure 4.28). Considering the principle axes of the ellipsoid as the optimum direction to generate angular momentum, the two configurations have different optimum directions. Both *forward/backward* and *twist* configurations give better manipulability than single sided configuration (*forward/forward*, *backward/backward*). Among all the configurations, *forward/backward* gives the best angular momentum manipulability.

Table 4.2 Angular Momentum Manipulability Related To Different Configurations

Manipulability	f/f	b/b	f/b	*twist*
ω_k	66.03	66.04	90.78	74.54
ω_{kb}	42.73	43.13	54.42	42.15
ω_{ka}	6.63	6.59	10.37	9.36

Note: $f \rightarrow$*forward*, $b \rightarrow$*backward*

Lift-up Motion

In the previous section, it has been concluded that the *forward/backward* configuration gives the best angular momentum manipulability. However the result only valid for the specific posture for which the corresponding CoM height is $0.8m$. In this part, the manipulability is going to be examined for a series of configurations. The robot is requested to do a lift up motion, the CoM height has been commanded from $0.7m$ to $0.88m$ (due to leg length limit). Angular momentum manipulability index ω_{ka} value has been recorded through out the whole process. Results for all four configurations are shown in the Figure 4.29. It is obvious that the *forward/backward* is the best for this motion among the four configurations. The *twist* configuration shows good manipulability with lower CoM height, and it decreases as the robot lift-up. The other two configurations have no difference for this motion. One might notes that the angular momentum manipulability index increases as the robot lift-up, this relationship is reversed for the linear one which decreases as the robot lift-up. This can be interpreted in the way that as the masses of the robot distributed further and further away from the CoM, they gain more and more influences on the centroidal angular momentum with increasing moment arms.

In general, the proposed configurations *forward/backward* and *twist* give better angular momentum manipulability for a wide range of postures. This is not the only benefits it brings to the robot, they also enables the new movement possibility: cross-step.

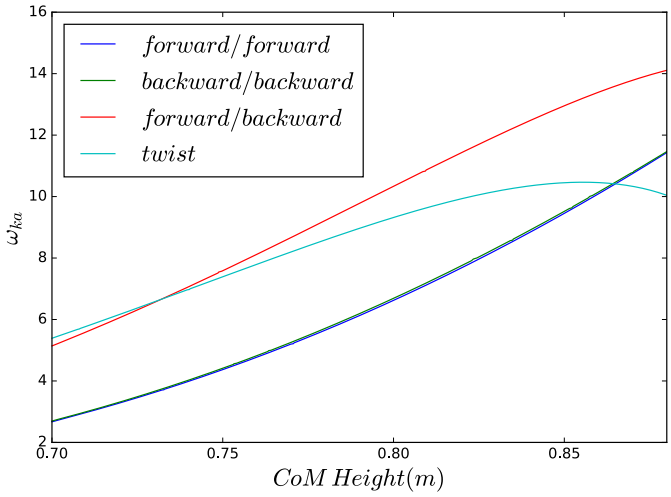


Fig. 4.29 Configuration related angular momentum manipulability through out the lift-up motion for four different configurations.

4.3.3 Cross-step Movements

Robust walking of humanoid robots often requires the robot to be adaptive to external disturbances and terrain irregularities. Three push recovery strategies that allow the robot to recover from different levels of external push has been presented in [91]. Push recovery stepping strategies have been proposed in multiple works [37] [139] [140] [141]. Additionally, different model predictive control (MPC) formulation have been proposed to updates foot placement online [118] [119] [120] [121]. To meet real-time requirement, linear models are often chosen as template model to perform iterative online optimization involved in the MPC control scheme. Non-linear formulation which involve step timing optimization have been explored in several recent studies [142][143][144]. Considering the worst case scenario in which the robot has been heavily pushed towards the right during the right support phase, a two step strategy is necessary: put down the left foot as close as possible to the right foot within as

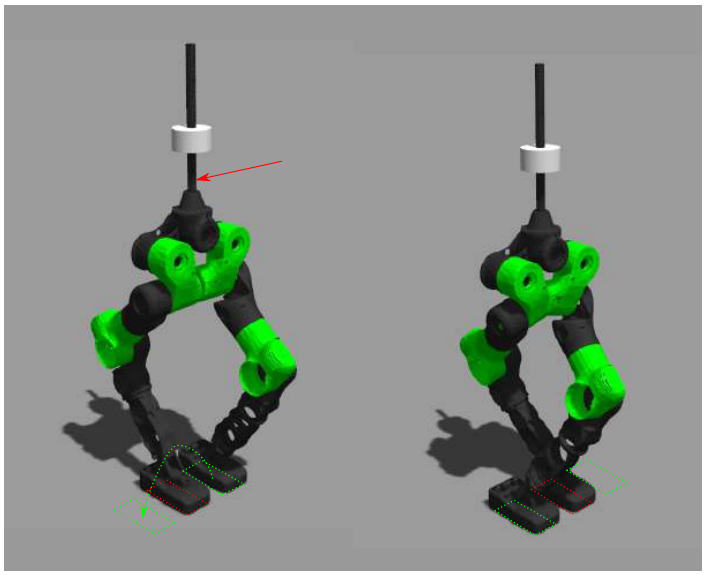


Fig. 4.30 Corss-step. The red arrow indicates a push force from left to right acting on the robot at a certain moment, and this initiates the cross-step action: the left foot swing over the right foot and lands on the right side of it.

short as possible duration, followed a large right side step. For human, a more natural reaction would be cross their legs to make a cross step directly. This action is however risky or mechanically impossible for most existing robots. The proposed configurations *forward/backward* or *twist* could be a solution to this problem.

The cross-step action with *forward/backward* configured robot is shown in the Figure 4.30. As can be seen from the figure, the robot switches from *forward/backward* configuration to *twist* configuration with one cross step. Actually, the same can happen from *twist* configuration to *forward/backward* configuration. As a result, the robot can switch between these two configurations infinitely which means that the robot can do multiple cross-steps continuously as plotted in Figure 4.31. One might notice that self-collision happens between the hip-pitch links, this is due to the fact that the

mechanical design is finished before we come up with this cross-step idea. But it is absolutely possible to avoid this problem with proper design.

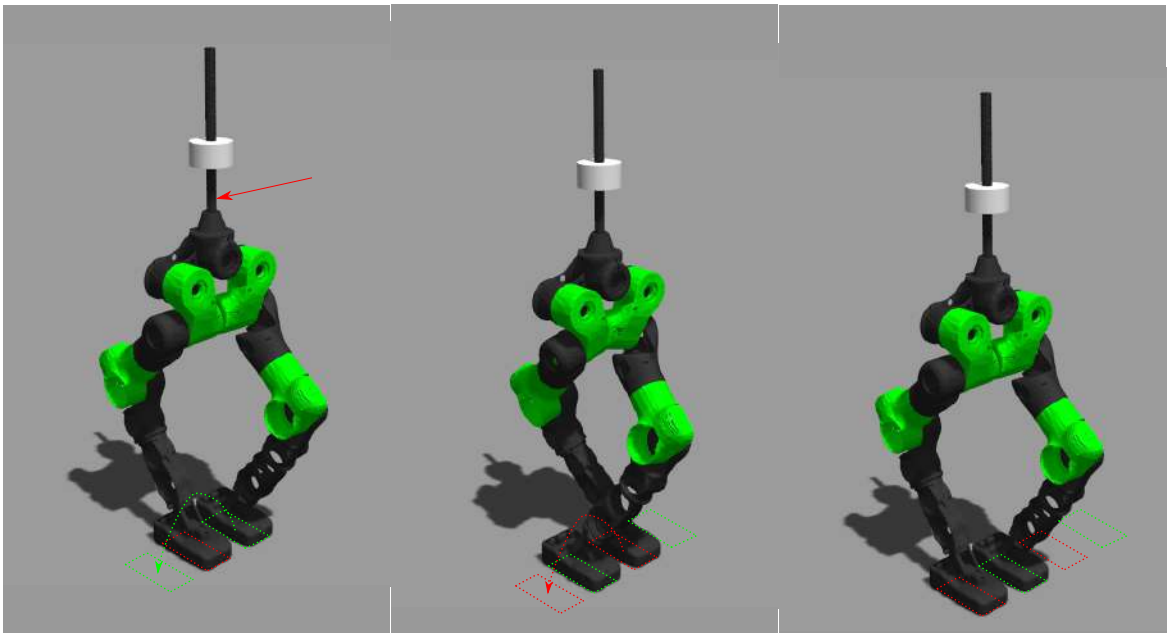


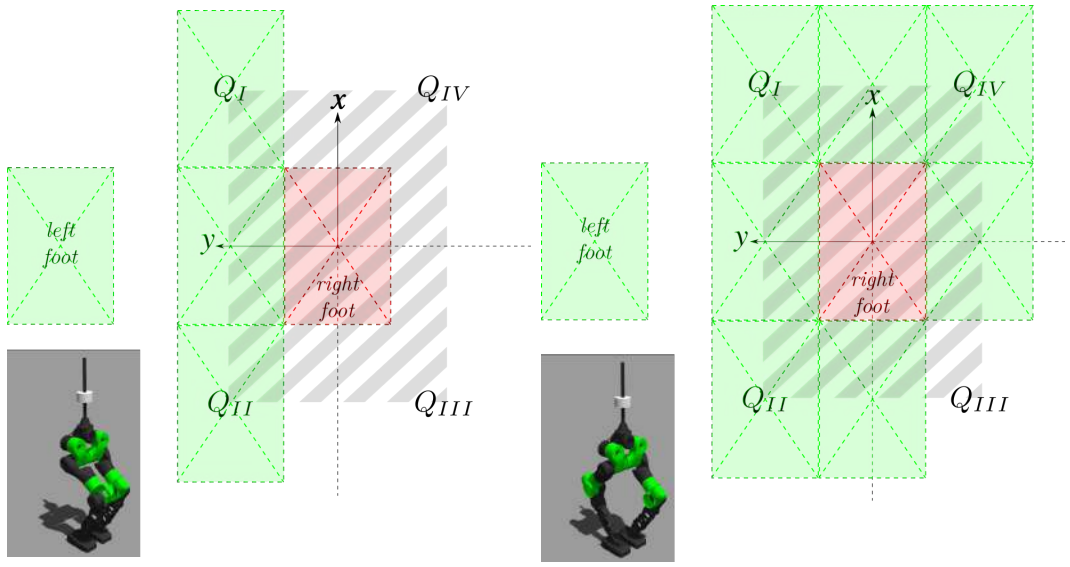
Fig. 4.31 Multiple corss-steps. The robot switches from *forward/backward* to *twist*, and then from *twist* to *forward/backward*. Footprints has been labeled with squares, the green ones represent the left footprints and the red ones are for the right foot.

Cross-step Feasible Region

With the possibility to do cross-step action, the feasible region for the swing foot is enlarged. For footstep planning, the feasible region $\mathcal{F}$ of the swing foot is usually defined by:

$$\mathcal{F} \in \mathcal{D} \cap \mathcal{K} \cap \mathcal{C} \tag{4.20}$$

where $\mathcal{D}$ is the design region, $\mathcal{K}$ is the kinematic feasible region, $\mathcal{C}$ is the collision-free region. For the two cases illustrated in Figure 4.32, they have different design region



(a) *forward/forward* (b) *twist*

Fig. 4.32 Design region for left swing foot while the robot takes different configuration. Assuming in right single support phase, the right stance foot has been plotted in red and a fixed frame has been attached to its center. The swing left foot is in green and several possible landing prints have been plotted for reference. Grey strip labels out unfeasible regions due to self-collisions between feet.

$\mathcal{D}$:

$$\mathcal{D}_{f/f} = \{(x, y) \in Q_I \cup Q_{II}\} \tag{4.21}$$

$$\mathcal{D}_{f/b} = \{(x, y) \in Q_I \cup Q_{II} \cup Q_{IV}\} \tag{4.22}$$

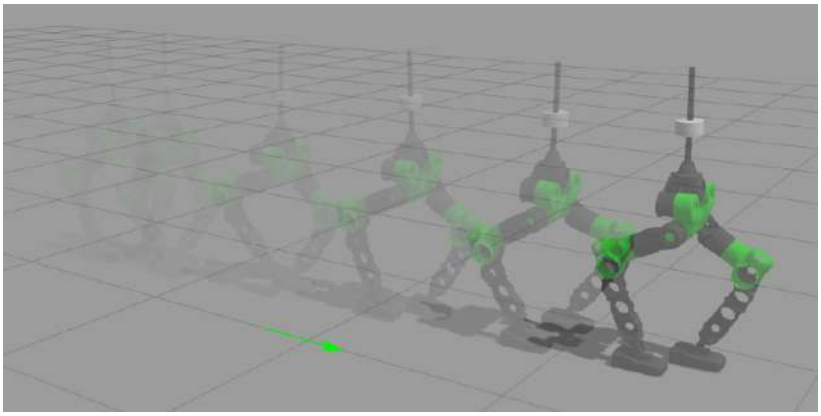
where Q_I, Q_{II} and Q_{IV} stands for quadrant I, II and IV of (x, y) plane, f/f and f/b stands for *forward/forward* and *forward/backward*. f/b configuration increased the design region with one more quadrant comparing to f/f configuration .

Simulation

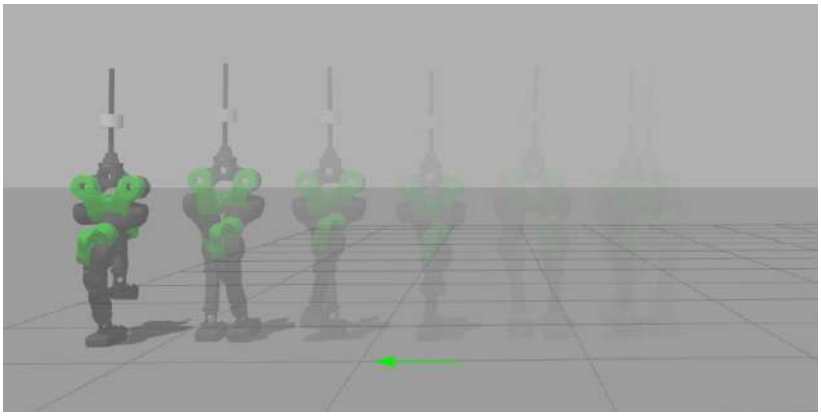
The walking motion is simulated in Gazebo + ROS (Robot Operating System) environment. For Gazebo simulator, ODE physics engine has been chosen. Each joint of the robot could be controlled in position mode or torque controlled. In the simulation, we choose pure torque control mode for all the joints. Robot Odometry data (pelvis position and velocity) and joint states (positions and velocities) has been sent to control system as state feedbacks. A two level hierarchical control system has been used to generate walking motion for the robot. High-level controller plans Cartesian space trajectories for CoM and feet. The low-level controller is a whole-body controller which takes the desired CoM and feet trajectories as input and finds out the joint-torques for all joints. The controller has been formulated as a quadratic optimization problem whose goal is to track desired trajectories as good as possible and at the same time with respect to all kinds of constraints, such as dynamic feasibility, friction cone, torque limits [120] [145]. The whole-body controller is running at a much higher frequency than the high level controller, in walking simulation 1000 Hz has been used.

Walking Planner

The model we used to generate the reference motion is linear inverted pendulum model (LIPM) and many walking pattern generation methods based on this model have been proposed [34] [35] [146]. The model is composed of a point mass and a massless telescopic leg. Therefore, the planner based on this model provide no information about the configuration of the robot. It only generates Cartesian space reference



(a) Walking Forward



(b) Walking Sideways

Fig. 4.33 Robot walking in different direction: (a) The robot is walking forward, (b) the robot is walking towards its right direction.

trajectories and it replans these trajectories at support switching moment for next three steps. Here it is assumed that no double support phase exists, left support phase switches to right support phase simultaneously. The planner is formulated as a liner model predictive control problem which takes future foot placements as optimization variables [120] [121]. Suppose there is a desired average velocity $\boldsymbol{v}_{ref}$ the robot needs to track. The first optimization goal is to minimize the least square error between this desired velocity $\boldsymbol{v}_{ref}$ and future switching moment CoM velocities (three steps have been considered in this work). The other goal is to minimize the least square errors between replanned footsteps and desired footsteps. Desired footsteps are calculated based on desired velocity $\boldsymbol{v}_{ref}$ with consideration of inter-feet clearance to avoid self collision between feet. With cross-step enabled, the feasible region of desired footsteps expanded as shown in (b) of Figure 4.32. However the stance foot makes the feasible region non-convex, in this case, we have to select a convex sub-region to make problem convex and solvable. In this paper, convex sub-region has been chosen in a heuristic way: either $Q_I \cup Q_{II}$ or Q_{IV} depending on how much lateral velocity been commanded or how much lateral disturbance been applied. Taking the lateral velocity as example, a threshold value could be defined in advance, if commanded lateral velocity goes beyond this threshold, the design region switch from $Q_I \cup Q_{II}$ to Q_{IV}. The same rule applies for lateral disturbance case.

Walking motion in different directions have been simulated as shown in Figure 4.33. Walking forward and walking sideways have been demonstrated. The robot takes cross-step action in the sideways walking. Actually, the cross-step action is trigged

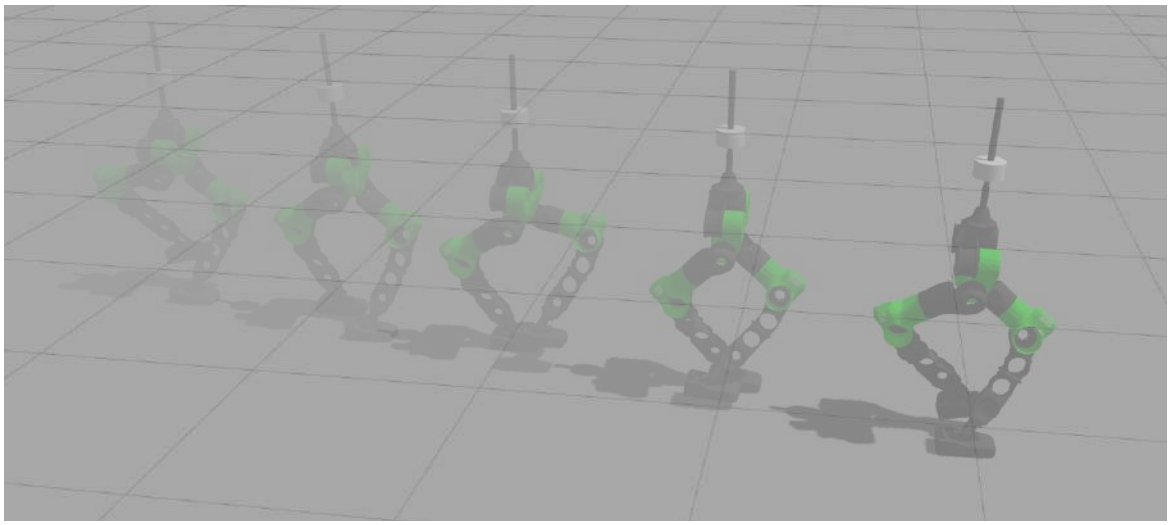


Fig. 4.34 Walking forward with *twist* configuration.

by the reference velocity $\boldsymbol{v}_{ref}$. with a y component of $\boldsymbol{v}_{ref}$ beyond certain threshold, it will trigger the cross-step motion for the robot. A small one will results in small side step without crossing legs. A strong side push on the robot could also trigger the cross-step in the same direction. One thing worth noting is that the robot does not have to switch back to *forward*/*backward* configuration to be able to walk forward. That is to say, the robot can perform walking forward motion in *twist* configuration as well and it is shown in Figure 4.34. This guarantees the robot could change walking direction at any stage of cross-step.

4.3.4 Conclusion

In this part, we propose two new configurations for humanoid balancing and walking. We have compared them with other regular configurations in terms of centroidal momentum manipulability. They indeed provide better angular momentum manipulability. One

major benefit of the proposed configurations is that they enable the robot to do cross-step motion. This is a useful skill for humanoid push recovery but long being ignored due to hardware limitation. With cross-step enabled, the robot is more robust to lateral direction disturbances. However, due to non-convex feasible region, traditional convex optimization can not be directly applied to plan cross-step. A simple heuristic has been proposed to overcome this problem. Walking simulation has been performed successfully to verify the proposed cross-step idea.

Chapter 5

Hopping and Running

To generate dynamic motions such as hopping and running on legged robots, model-based approaches are usually used to embed the well studied spring-loaded inverted pendulum (SLIP) model into the whole-body robot. In producing controlled SLIP-like behaviors, existing methods either suffer from online incompatibility or resort to classical interpolations based on lookup tables. Alternatively, this paper presents the application of a data-driven approach which obviates the need for solving the inverse of the running return map online. Specifically, a deep neural network is trained offline with a large amount of simulation data based on the SLIP model to learn its dynamics. The trained network is applied online to generate reference foot placements for the humanoid robot. The references are then mapped to the whole-body model through a QP-based inverse dynamics controller. Simulation experiments on the WALK-MAN robot are conducted to evaluate the effectiveness of the proposed approach in generating bio-inspired and robust running motions.

5.1 Heuristic Controller

In the early stage of legged robot running research, heuristic controllers has been developed to realize hopping and running motion on one, two or four legged robots. The controller is developed relying on deep intuition of the system dynamics, applying separate control laws to different parts of the locomotion task. With such kind of decomposition, the controller becomes simple but yet effective. Mainly three independent sub-tasks are involved in the controller: hopping height, body-posture and forward velocity. First applied to monopod, this method has been later extended to 2D and 3D biped. But all those robots share similar design of light prismatic legs which makes them unrealistic when compared to articulated humanoid robot. In this part, we are going to apply the same controller to our humanoid robot and it will be the base line for comparison with our newly proposed neural-network controller in the next section 5.2.

Hopping Height As described in [147], the hopping motion is an oscillation governed by the mass of the body, the springiness of the leg and the gravity. These character features are summered from the light-weighted prismatic leg design of their hopper. For our humanoid robot, they can not be directly applied but equivalences exist. The springy prismatic leg could be simulated with a virtual spring that connects the CoM of the robot and the feet as shown in Figure 5.1.

One part of the original control system is to excite cyclic hopping motion while regulating the height. It indicates that the control law for this task should also consists

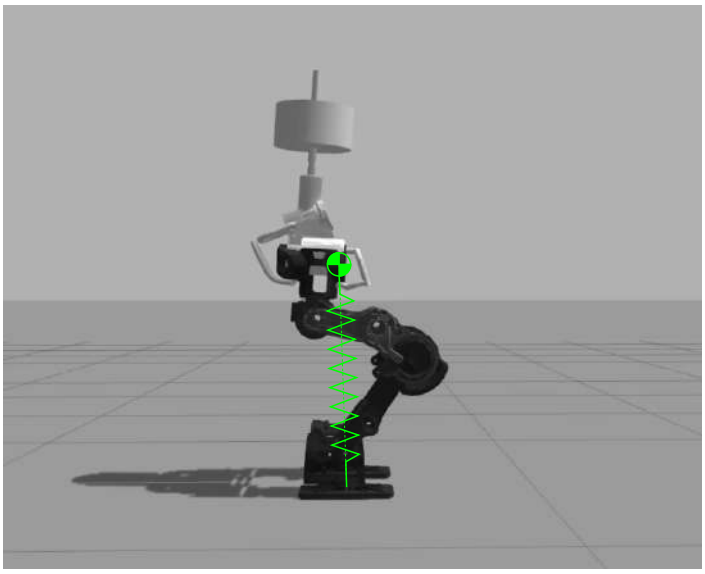


Fig. 5.1 Virtual spring connecting CoM and feet.

two parts: exciting cyclic motion part and height regulation part. For the exciting part, the robot needs to generate spring force in the virtual spring axial direction. The magnitude of the force can be calculated based on virtual spring length difference between current one and set point. The direction of the force is from the feet to the CoM. For implementation, virtual model control method could applied to incorporate the virtual spring force into the whole-body controller. With only the exciting part, the hopping motion is basically emulating the idea mass-less spring behavior. For realistic simulation considering contact model, this is not going to last forever since the system energy will be lost gradually due to landing impacts. That's why we need another regulation part to inject energy and sustain the hopping motion. This can be achieved in different ways. The original plan is to deliver a vertical thrust during each support phase and this can be achieved with controlled hydraulic cylinder. Here, we would like to achieve the same goal with proper virtual spring extension. A simple

proportional controller that calculates spring extension based on the hopping height error would work: $\Delta l = k_P(h_{des} - h_{last})$.

Body Posture The body posture control part is just to keep the upper body of robot upright. The original plan is to use a simple PD controller to regulate the body posture towards desired orientation. Since the hopper has a point feet, it is actually utilizing the contact force (mainly the tangential friction force) to help orienting the torso. For a humanoid robot with a planar feet, this task is essentially easy since extra torque could be acquired even through it is limited by the size of the feet. This task can be also easily considered in the whole-body controller as an orientation regulation task.

Forward Speed The key to maintaining long-term balance and control the global motion is foot placement. Forward speed can be regulated by foot placement too. The original proposed control law is to place the leg to an appropriate forward position with respect to the body during the flight phase. The foot placement has strong influence on the ensuing stance phase, depending on where it is, the robot could accelerate or decelerate. There exists a neutral foot position which would results in symmetric stance phase and therefore zero net acceleration. Then the control of forward speed is just a matter of linear corrections of foot position with respect to this neutral point. For the humanoid robot, a slightly modified version is adopted:

$$x_f = k_1 \frac{\dot{x} T_{st}}{2} + k_2(\dot{x} - \dot{x}^*) \tag{5.1}$$

where x_f is the calculated foot placement in x direction with respect to CoM, T_{st} is the expected stance time (previous stance period could be used), $\dot{x}$ and $\dot{x}^*$ stands for current and desired CoM velocity, k_1 is the neutral point correction gain and k_2 is velocity correction gain. $k_1 \dot{x} T_{st}/2$ is an estimated neutral point, on top of it the correction term $k_2(\dot{x} - \dot{x}^*)$ has been added to gradually correct the velocity. Of course, a dead-beat controller is possible to bring the robot to desired velocity in less steps, but it requires a lot of simulation or experimental data to build up the loop-up table.

This control law is applied during the flight phase. As mentioned before, although the angular momentum is conserved during the flight phase, the relative foot placement still could be controlled. Knowing the CoM velocity during the flight phase, foot placement could be constantly corrected. In this way the robot becomes reactive to external disturbance even in flight phase. We call it in step correction. However, the velocity information is difficult to get only with on board sensors in flight phase. Assuming that the CoM horizontal velocity in flight phase does not change, the velocity at previous take off moment could be used in (5.1). This implementation may results in a less responsive controller but step to step correction is still enabled.

5.2 Neural-Network Controller

The Spring-Loaded Inverted Pendulum (SLIP) model is a well recognized template model [148] for hopping and running based on the biomechanical studies [149, 150]. Despite its simplicity, it accurately describes the CoM dynamics, ground reaction force

profiles, and transitioning between different phases of motion observed in humans [151]. Thanks to its reductive and platform-independent model, it has been widely used in the design and control of legged robots [152–155]. In particular, the controlled SLIP is used as a planner in the high-level control structures of robot to provide it with references that are implicitly consistent with the natural dynamics of running.

The SLIP running is a dynamic gait rendering cyclic stability, which requires a sufficiently large prediction horizon for control. Early studies in this regard are largely influenced by the simple intuitive control implemented on Raibert's hoppers [147]. The machines were able to exhibit dynamic behaviors while it was assumed that the control of hopping height, speed and posture are decomposed. Although inspiring, all those robots share similar design of light prismatic legs, i.e., a SLIP-like morphology. When it comes to controlling legged robots with non SLIP-like morphologies like a humanoid, such an intuitive approach inspired by biology alone shows limited success. Moreover, the aforementioned decoupling leads to long convergence time due to the simplistic model used for control.

A large body of research in the SLIP literature has been directed towards more accurate and realistic controls, most of which may be categorized into two schemes: the methods which implement dead-beat like controllers through solving the running return maps [156, 42, 44, 157]; and tabular control methods relying on look-up tables constructed upon the data generated by comprehensive forward-in-time simulations covering a wide range of SLIP states and parameters [158–160]. Application of the former to online control is not preferred, due to the non-linear optimization inevitably

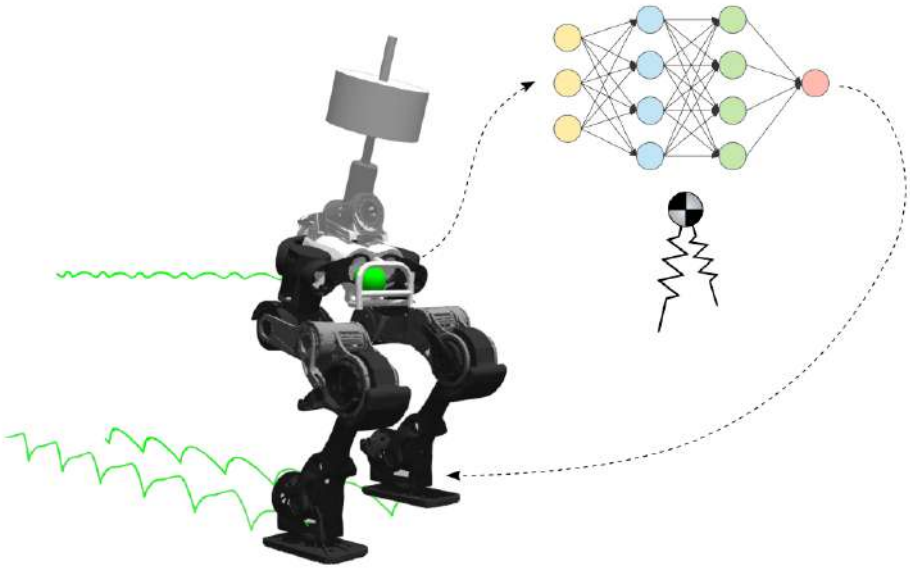


Fig. 5.2 WALK-MAN Lower body running in simulation. The Center of Mass dynamics of the robot is controlled to match that of a SLIP model. A neural network trained offline with a large amount of SLIP simulation data is used to encode the foot placement behaviour.

involved in the computations. The latter is fast enough for online implementation since a look-up table can be constructed offline. However, it is practical only for the range of parameters using which the look-up table is constructed. Moreover, the size of the table grows exponentially with the number of the input variables, which challenges the generality of the approach.

The present work strives to fill the gap between the non-linear optimization method and the classical look-up table method by using a deep neural network. The network is trained offline with large amount of simulation data based on the SLIP model to learn its dynamics. Once this most time-consuming part has been done, the trained network could be easily deployed online for real-time querying. The knowledge learned from simulation data are encoded in a limited number of weight parameters and this

parametric representation does not enlarge with inputs and outputs. Comparing to the look-up table approach, the interpolation between data are naturally embedded inside the network.

Having developed an effective data-driven controller for the SLIP planner, we consider the embedding of the template behaviors into the whole-body robot. At this stage we need to focus also on the abstracted out dynamics such as the touchdown impact and torso stabilization. The former is known to be a real challenge in control of legged robots in general and dynamic gaits in particular, as the impact phenomenon is a fuzzy phase of motion. In order to make sure that the planned template behaviors, which are intentionally constrained to be energy conservative, remain feasible for the real robot, we adopt an energy regulation technique that determines the takeoff moment as the moment at which the energy lost due to the touchdown impact has been restored. Application of this takeoff event condition on top of the previously proposed leg length modulation [161] remarkably improves the robustness against uncertainties introduced by the touchdown impact and other unmodeled dynamics in the planning phase. This takeoff event modulation together with the CoM reference trajectory and the touchdown angle tracking, all planned by the controlled SLIP, are mapped into the whole-body robot through a state-of-the-art QP-based inverse dynamics controller.

5.2.1 Spring Loaded Inverted Pendulum Model

The spring-loaded inverted pendulum (SLIP) model consists of a point mass m and a massless spring with stiffness k and rest length l_0 as shown in Figure 5.3. Three

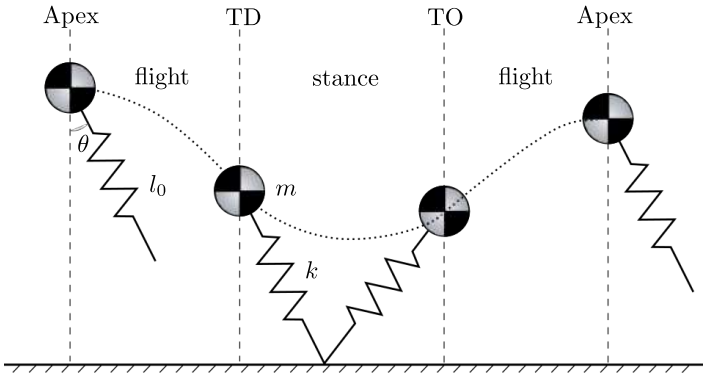


Fig. 5.3 The spring-loaded inverted pendulum (SLIP) model. The figure shows the sequence of events (Apex-TD-TO-Apex) and phases (flight-stance-flight) involved in one step of running. The leg angle at TD moment (θ_{TD}) decides the evolution of stance and ascending flight phases.

phases (flight-stance-flight) are involved in one step of the running motion and they are separated by touchdown (TD) and takeoff (TO) events. During flight phase, the mass follows a ballistic projectile trajectory with the dynamics:

$$\begin{cases} \ddot{x} = 0 \\ \ddot{z} = -g \end{cases} \tag{5.2}$$

where (x, z) are the coordinates of the point mass in sagittal plane, and g is the gravitational acceleration. The massless leg can be arbitrarily positioned during flight phase in preparation for the touchdown. At the touchdown moment, the system switches to stance phase, the point mass follows the dynamics:

$$\begin{cases} m\ddot{x} = k[l_0(x^2 + z^2)^{-1/2} - 1]x \\ m\ddot{z} = k[l_0(x^2 + z^2)^{-1/2} - 1]z - mg \end{cases} \tag{5.3}$$

Once the leg length $l = \sqrt{(x^2 + z^2)}$ reaches its rest length l_0 during spring extension, the system takes off and enters the flight phase again.

The apex point ($\dot{z} = 0$) during flight phase is usually chosen to study the periodic motion of system. At the apex point, the system state can be described by one variable $\dot{x}$ (or z) due to the total energy conservation:

$$\frac{1}{2}m\dot{x}^2 + mgz \equiv E \tag{5.4}$$

where E is the total energy of the system which is conserved throughout the whole process. Here the velocity $\dot{x}$ at apex point is chosen since we are more interested in regulating the running speed. Given the speed at one apex point, the system behavior in the ensuing stance and flight phases is fully determined by the touchdown angle θ_{TD}. The next apex state is a function of current apex state and the touchdown angle:

$$\dot{x}_{n+1} = f(\dot{x}_n, \theta_{TD,n}) \tag{5.5}$$

where n denotes the current running step. A one step deadbeat controller emerges by inverting this apex return map:

$$\theta^*_{TD,n} = f^{-1}(\dot{x}_n, \dot{x}^*_{n+1}) \tag{5.6}$$

where $\theta^*_{TD,n}$ is the touchdown angle that ensures reaching the desired velocity $\dot{x}^*_{n+1}$ at the next apex. However, the hybrid nature of the return map and nonlinearity of stance

phase dynamics (5.3) exclude the possibility of finding a closed form solution for this inverse relationship. As such, the problem of finding $\theta^*_{TD,n}$ is inevitably transformed into a nonlinear optimization problem:

$$\begin{aligned} \theta^*_{TD,n} &= \min_{\theta} ||\dot{x}^*_{n+1} - f(\dot{x}_n, \theta)||^2 \\ s.t. \quad & \theta_{min} < \theta < \theta_{max} \end{aligned} \tag{5.7}$$

where θ is the touchdown angle to be optimized to bring the system state at next apex $f(\dot{x}_n, \theta)$ as close as possible to the desired one $\dot{x}^*_{n+1}$ respecting the angle limits. Usually, this time-consuming optimization process can only be conducted offline, while convergence cannot be guaranteed. These limitations motivate us to explore a different possibility which better suits the online implementation requirement, that is a neural-network-based representation for the inverse mapping (5.6).

5.2.2 Deep Neural Network Controller

The proposed neural network takes the inputs $[\dot{x}_n, \dot{x}^*_{n+1}]$ and outputs the touchdown angle $\theta^*_{TD,n}$. A deep learning techniques is adopted to train the network offline. The trained network is then applied online which produces an output for every possible inputs. Below, we first describe how valid datasets are generated for training the network and then present the structure of the neural network and the training process in detail.

Data Generation

The neural network under consideration learns from datasets that comes from apex-to-apex simulations of the SLIP model as given in (5.5). Each simulation produces one dataset. For simplicity, we choose a constant energy level and all simulations are performed with this energy level E_{cons}. Given a fixed energy level, the initial state can be completely determined by an initial horizontal velocity $\dot{x}_0$. Together with a touchdown angle θ_0, we can simulate forward the SLIP model to get the next apex velocity $\dot{x}_1 = f(\dot{x}_0, \theta_0)$. At this point, a training example has been generated. A general representation of this process is:

$$\dot{x}_1^{(i)} = f(\dot{x}_0^{(i)}, \theta_0^{(i)}) \tag{5.8}$$

from which a training example $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ is collected as:

$$\begin{cases} \boldsymbol{x}^{(i)} = [\dot{x}_0^{(i)}, \dot{x}_1^{(i)}]^T \\ \boldsymbol{y}^{(i)} = [\theta_0^{(i)}] \end{cases} \tag{5.9}$$

where $i = 1, 2, ..., n$. Repeating this process with a different initial velocity and touchdown angle, the whole data set can be collected:

$$\boldsymbol{X} = \begin{bmatrix} -(\boldsymbol{x}^{(1)})^T- \\ -(\boldsymbol{x}^{(2)})^T- \\ \cdots \\ -(\boldsymbol{x}^{(n)})^T- \end{bmatrix} \qquad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}^{(1)} \\ \boldsymbol{y}^{(2)} \\ \cdots \\ \boldsymbol{y}^{(n)} \end{bmatrix} \tag{5.10}$$

One thing worth mentioning is that the initial velocity $\dot{x}_0^{(i)}$ and touchdown angle $\theta_0^{(i)}$ can be chosen randomly but should be within reasonable limits. Specifically, touchdown angle limits are defined as: $\theta \in [0, \tan^{-1}(\mu)]$ where μ is the static friction coefficient. The velocity is limited in the range $[0, \sqrt{2(E_{cons} - mgl_0)/m}]$ where the upper bound is defined with respect to the minimum height (rest length l_0), the slip model can take. Below is the pseudocode used to generate the training data sets. To be concise,

$\boldsymbol{X}, \boldsymbol{Y} = [], [];$
for *simulation (i=0, i<n, i++)* **do**
  $\dot{x}_0^{(i)} = rand(0, \sqrt{2(E_{cons} - mgl_0)/m});$
  $\theta_0^{(i)} = rand(0, \tan^{-1}(\mu));$
  $\dot{x}_1^{(i)} = f(\dot{x}_0^{(i)}, \theta_0^{(i)})$;
  $\boldsymbol{x}^{(i)} = [\dot{x}_0^{(i)}, \dot{x}_1^{(i)}]^T$;
  $\boldsymbol{y}^{(i)} = [\theta_0^{(i)}]$;
  $\boldsymbol{X}$.insert$(\boldsymbol{x}^{(i)})$;
  $\boldsymbol{Y}$.insert$(\boldsymbol{y}^{(i)})$;
end

Algorithm 1: Generating the training data sets

we have not presented the guard functions inside the forward simulation we used to eliminate those bad data examples such as certain combination of $\dot{x}_0^{(i)}$ and $\theta_0^{(i)}$ which

leads to negative vertical velocity at TO moment. In this paper, the parameters used for training is $E_{cons} = 750, m = 85, l_0 = 0.8, k = 42500$. A training set of size 30000 is collected and 5% of it has been used as test set.

Neural Network Structure and Training

To learn the generated data, a fully-connected feed-forward network (FNN) has been used. Compared to tabular approaches [158] which check the entry closest to a given input and produce the associated output, FNN allows to generalize to different inputs. In addition, it can model non-linear functions by using non-linear activation functions, improving over the previous linear methods [162]. For a given input $\boldsymbol{x}^{(i)}$, we can define the FNN in a recursive way as follows:

$$\begin{gathered} \boldsymbol{h}_l = f_l(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l), \quad \forall l \in \{1, \cdots, L\} \\ \text{with} \quad \boldsymbol{h}_0 = \boldsymbol{x}^{(i)} \quad \text{and} \quad \boldsymbol{h}_L = \hat{\boldsymbol{y}}^{(i)}, \end{gathered} \tag{5.11}$$

where L is the total number of layers, f_l is the activation function applied on the corresponding layer l, $\boldsymbol{W}_l$ are the weight matrices, $\boldsymbol{b}_l$ are the bias terms, and $\hat{\boldsymbol{y}}^{(i)}$ is the predicted output. We can summarize the above equation by $\hat{\boldsymbol{y}}^{(i)} = f_{NN}(\boldsymbol{x}^{(i)}; \boldsymbol{W})$ where $\boldsymbol{W} = \{\boldsymbol{W}_1, \boldsymbol{b}_1, \cdots, \boldsymbol{W}_L, \boldsymbol{b}_L\}$ are the weights that need to be optimized. In our experiments, our network has 3 hidden layers with 20, 50, and 20 units respectively. We used 'relu' as the non-linear activation function for each hidden layer. To avoid overfitting, we regularize our network using dropout. The training was carried out

using the mean-squared loss:

$$\mathcal{L}_{MSE} = \sum_{i=1}^{N} ||\boldsymbol{y}^{(i)} - f_{NN}(\boldsymbol{x}^{(i)}; \boldsymbol{W})||^2, \tag{5.12}$$

along with the Adam optimizer. We trained the network for 100 epochs using a batch size of 128, and a learning rate of 0.0001.

5.3 Mapping SLIP-like Motions to Whole-body Robot

We now describe how the planned template behaviors are encoded into the whole-body robot. The humanoid robot used for the simulations is the WALK-MAN [163]. In simulation we only use the lower-body of the robot for simplicity but with the idea in mind that the upper body could improve the performance by fully utilizing the swing motion of arms [44].

5.3.1 SLIP model to Whole-body Model

The fundamental difference between the template and real robot models is the inertia distribution. Different from the SLIP model which can arbitrarily position its mass-less leg during the flight phase, the swinging movement of heavy legs of our humanoid robot would cause noticeable change in the torso orientation. One needs to carefully consider this in designing the swing leg movement, otherwise the robot can shortly lose the balance. We avoid this problem by limiting the movement speed of swing leg at a cost of velocity regulation speed.

Apart from that, the effect of touchdown impact for the robot with heavy legs is severe, and it results in considerable energy loss in each step. Since our SLIP model simulation is conducted at a certain energy level (5.4), it is critical to maintain the same energy level for the robot to ensure relevant mapping. A number of strategies are proposed to compensate the energy loss. In [161], an energy correction law is proposed to inject energy with pre-compressed spring leg before touchdown based on a pre-computed value of the energy loss assuming an ideal inelastic impact.

$$\begin{aligned} \Delta E_{loss} &= \frac{1}{2}|(\dot{\boldsymbol{q}}^+)^T \boldsymbol{H}(\dot{\boldsymbol{q}}^+) - (\dot{\boldsymbol{q}}^-)^T \boldsymbol{H}(\dot{\boldsymbol{q}}^-)| \\ &= \frac{1}{2}(\dot{\boldsymbol{q}}^-)^T \boldsymbol{J}_c^T (\boldsymbol{J}_c \boldsymbol{H}^{-1} \boldsymbol{J}_c^T)^{-1} \boldsymbol{J}_c (\dot{\boldsymbol{q}}^-) \end{aligned} \tag{5.13}$$

The resting length of the spring at touchdown is changed to:

$$l_0^+ = l_0^- + \Delta l \tag{5.14}$$

where $\Delta l = \sqrt{2\Delta E_{loss}/k}$. In practice, we find it difficult to compute an accurate estimation of the energy loss due to the touchdown timing inaccuracy and impact model mismatch. The impact is usually modelled as an elastic contact lasting for a period grater than an instant. As such, techniques relying merely on feed-forward calculations such as the one presented above may not guarantee an appropriate energy regulation. To tackle this issue, we propose a touchdown energy boost up and takeoff energy cut-off action pair. At TD instant, extra boost up energy ΔE_{boost} has been

added so as to make sure that the system will achieve higher energy level when the leg extends to reach

$$l_0^+ = l_0^- + \Delta l + \Delta l_{boost} \tag{5.15}$$

where $\Delta l_{boost} = \sqrt{2\Delta E_{boost}/k}$ is the extra extension due to boost energy. However, for the TO detection, instead of checking leg length, energy checking will be used to decide the TO moment:

$$TO : E = \frac{1}{2}mv^2 + mgz > E_{cons} \tag{5.16}$$

where E_{cons} is the desired energy level, beyond this level, the robot switches to flight phase immediately. Applying this method on the humanoid robot, we are able to bring the system energy to desired level in one step.

5.4 Simulation

The simulation environment used in this paper is Gazebo + ROS (Robot Operating System). The default physics engine ODE (Open Dynamics Engine) has been chosen to simulate the whole-body robot. The control loop runs at 1 kHz and control commands are sent to Gazebo through ROS. Two dimensional sagittal plane hopping has been simulated first to verify the proposed method. After that, running in three dimensional space has been conducted by composing two template models [164].

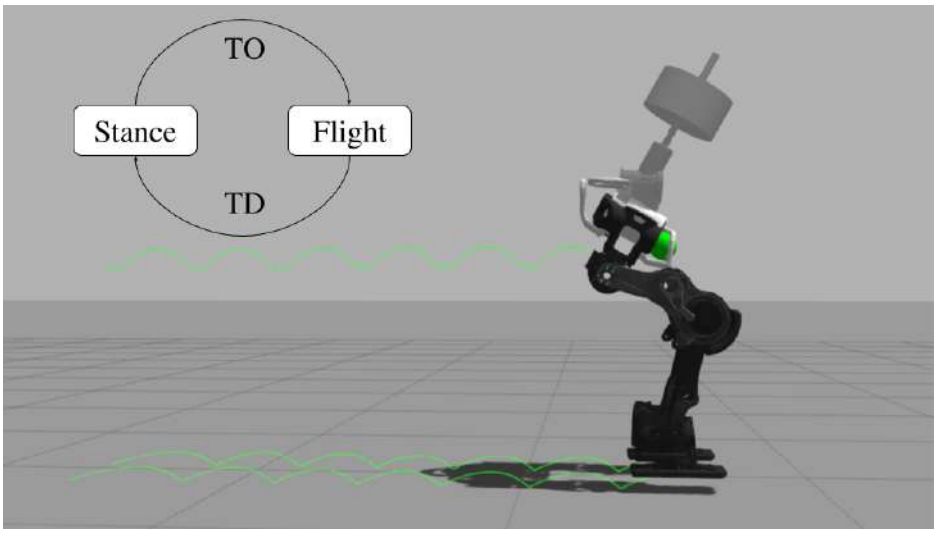


Fig. 5.4 Snapshot of sagittal plane hopping. The green sphere represents the CoM of the whole-body robot. The three green traces show the history of CoM and center points of the feet. Two states (Stance and Flight) and two events (TD and TO) are involved.

5.4.1 Hopping in Sagittal Plane

The hopping motion in sagittal plane can be well approximated by a two dimensional SLIP model. Two phases involved in the motion: stance phase and flight phase. A state machine has been employed to monitor the phase transitions. It is constantly checking TO or TD event to trigger the corresponding transition. TD happens when the feet touch the ground. Different checking methods (feet height, force-torque sensor) may trigger the transition at varying moment. Thanks to the energy boost and cut-off (5.15) (5.16) correction, the robot could end up with the same TO energy level.

Tasks involved in each phase are different. The whole-body controller will switch between the various tasks based on the state machine.

During the stance phase, tasks being closely tracked are: centroidal dynamics, torso orientation regulation and ground reaction force distribution (equal distribution

between the feet). For the feet, no specified goals are given, they adapt to the ground due to the non-slip constraints. At the TD moment, the high level controller will forward simulate once with the SLIP model (rest length modified for energy injection purpose) to get the whole CoM trajectory during the following stance phase. This CoM trajectory is then been used as desired tracking trajectory for the whole-body controller.

During the flight phase, the robot is under-actuated. The system CoM follows a ballistic trajectory. The only task to be controlled is the foot placement. In this work, no trajectories are specifically designed for the feet. The foot placement targets with respect to CoM are calculated from the touchdown angle provided by the trained neural network:

$$\begin{cases} {}^{G}x_f = l_0 \sin(\theta_{TD}) \\ {}^{G}z_f = l_0 \cos(\theta_{TD}) \end{cases} \tag{5.17}$$

For accurate velocity tracking, on top of the touchdown angle provided by the trained neural network, an simple PID controller has been added:

$$\begin{aligned} \theta_{TD} &= \theta_{ff} + \theta_{fb} \\ &= f_{NN}([\dot{x}, \dot{x}^*]) + \mathrm{PID}(\dot{x} - \dot{x}^*) \end{aligned} \tag{5.18}$$

where θ_{TD} is the reference touchdown angle sent to the whole-body controller which is composed of a feed-forward term θ_{ff} and a feedback term θ_{fb}. $\dot{x}$ is the current CoM velocity and $\dot{x}^*$ is the desired one during the next flight phase.

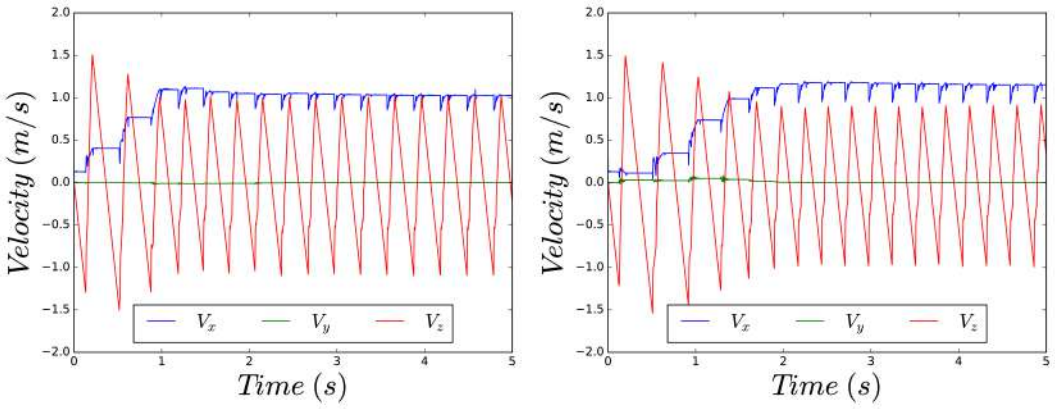


Fig. 5.5 Sagittal plane hopping CoM velocity. On the left side is the results from the neural network controller, and the right one comes from the Raibert controller. In both cases, the robot are released from the same height of 0.85 m with a forward velocity of 0.3 m/s. The target velocity is 1 m/s.

For this sagittal plane hopping case, the goal is to regulate the forward speed to 1.0 m/s. The robot is released from a 0.85 m height in the air and with an initial velocity of 0.3 m/s in x direction. It directly enters flight phase after releasing. These initial states are chosen rather randomly without special calculation. The CoM velocity recorded from the simulation is plotted in Figure 5.5. As a comparison, we also plot the data recorded from another simulation in which the Raibert foot placement controller has been used [17]. Theoretically, the SLIP model should be able to regulate to any achievable velocity in one step. However, this ability is limited by the touchdown angle range and also by the kinematic limits and actuation limits presented in the whole-body robot. In spite of that, it can be seen that the neural network controller took fewer steps to reach the desired velocity and also with less regulation error. In both cases, the energy correction law successfully regulate the energy to the desired level within one step. The results also prove its generality.

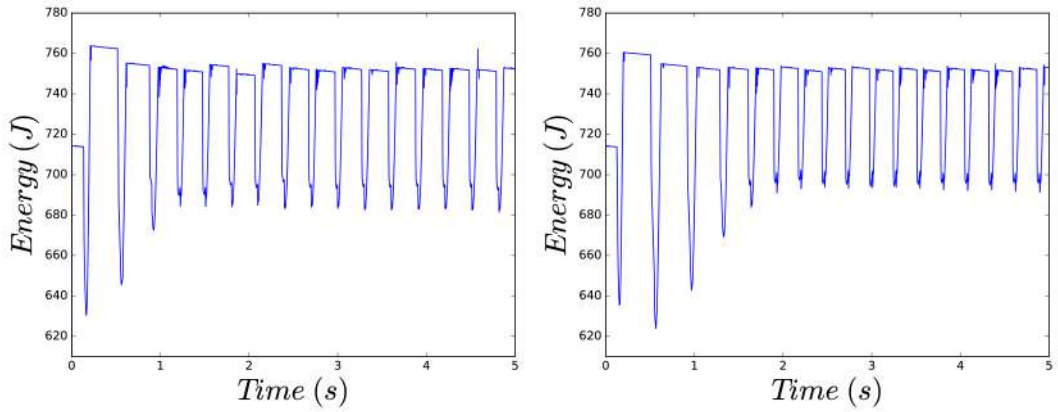


Fig. 5.6 Sagittal Plane Hopping Energy Level. Results come from different controllers but the same energy regulation law. The left one is our neural network controller and the right one is the Raibert controller

5.4.2 Running in 3D

Running is a three dimensional movement. In the previous section, the two-dimensional case has demonstrated the effectiveness of the neural network controller. To extend it to three dimensional space, two possibilities are: 1) composing two two-dimensional SLIP model to generate three dimensional running. 2) considering the 3D-SLIP model. The later one requires generating new simulation data and train a new neural network with expanded input and output dimensions. In this paper, we adopt the former idea. Without any modification of the trained neural network, we directly apply it to the lateral plane motion. Actually, observing the training data, a large amount of simulation ends up with reversed TO velocity comparing to the initial velocity which is exactly the case of lateral hopping motion. Therefore, running is treated as a hopping motion composed of sagittal hopping and lateral hopping, each component is governed

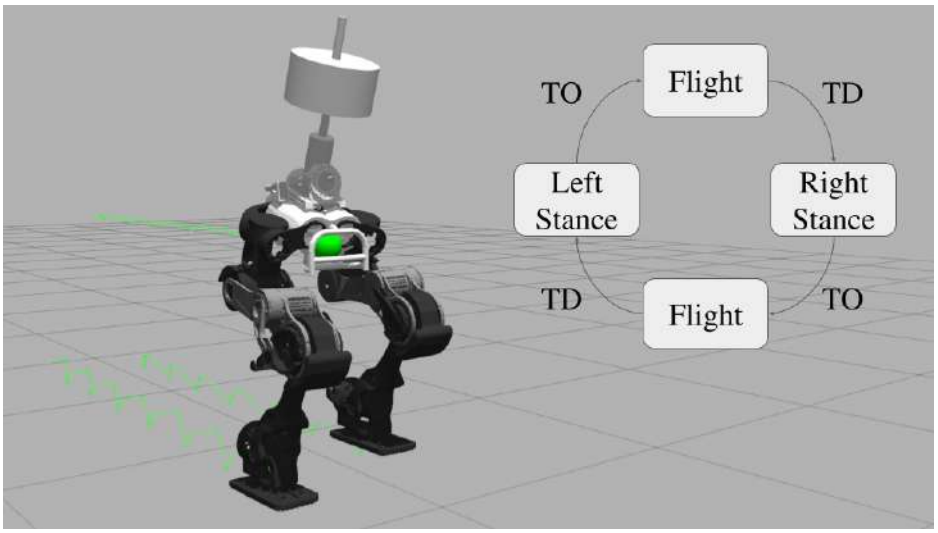


Fig. 5.7 Snapshot of running. The green sphere stands for the CoM position of the robot. The three green traces represent the history of CoM and center points of the feet. State machine includes three states: Flight, Left Stance, Right Stance. Two events (TO and TD) trigger the transition between these states.

by a two-dimensional SLIP model. These two template models are synchronized by a state machine as shown in Figure 5.7.

Tasks controlled in single stance phase are: centroidal dynamics, torso orientation and swing foot placement tracking. For the stance foot, no specified goals are given, it adapts to the ground due to the non-slip constraints. No ground reaction force distribution is needed since all forces comes from the single stance foot. In flight phase, only the foot preparing for landing is paid more attention to and it is controlled carefully to track the touchdown angle provided by neural network. The other foot stays in idle mode in horizontal direction and only keeps a clearance between itself and the ground. Additionally, any foot in the air is always controlled to be parallel to the ground.

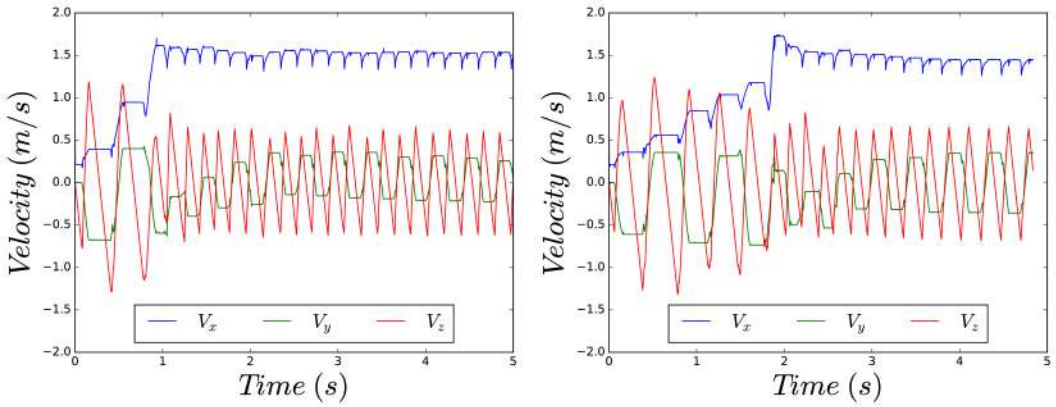


Fig. 5.8 Running CoM velocity plot. Results from composed neural network controller (left) and composed Raibert controller (right). The robot is released from the height of 0.85 m with a forward velocity of 0.2 m/s. After three steps it regulates to desired velocity 1.5 m/s.

Again, the CoM velocity results are compared between the neural network controller and the Raibert controller are shown in Figure 5.8. The neural network controller shows better regulation speed and less steady state error.

5.5 Extension to 3D-SLIP

Previously, a neural-network controller has been build based on 2D-SLIP model. For the 3D running, we composed two 2D-SLIP models to realize it. A more straight forward way would be directly derive the controller from 3D-SLIP model. Since the neural-network is flexible for input and output dimension, it is easy to extend the 2D case to 3D. Here we briefly explains how the same formulation could be extended to 3D. As shown in Figure 5.9, there is one more parameter to parameterize the swing leg orientation in this case. The touchdown angle is still θ and ϕ is a splay angle which is the angle between swing leg projection and x axis direction. Other parameters such as

m, k and l_0 have been considered as constants and they stands for mass, stiffness and rest length of the model.

The dynamics in flight phase is: $\ddot{\boldsymbol{p}} = \boldsymbol{g}$, where $\boldsymbol{p}$ stands for the position of the mass and $\boldsymbol{g}$ is the gravity vector. In stance phase, the dynamics becomes:

$$m\ddot{\boldsymbol{p}}_c = k(l_0 - ||\boldsymbol{l}||)\hat{\boldsymbol{l}} + m\boldsymbol{g} \tag{5.19}$$

where l_0 is the rest length of the model, $\boldsymbol{l} = \boldsymbol{p} - \boldsymbol{p}_f$ is the length of the spring in stance phase.

These dynamics are characterized in inertial coordinates. However for the data generation, local coordinates has been chosen to simplify the parametrization involved in the training process. In a local frame, the initial state at current apex point, the slip model could be fully described by its height z_n and forward velocity $\dot{x}_n$. After one step, at the next apex point, the state could be characterized by z_{n+1} and horizontal velocity $[\dot{x}_{n+1}\ \dot{y}_{n+1}]^T$. Further more, the magnitude of the horizontal velocity is coupled to the height z_{n+1} with constant energy E with $\dot{x}_{n+1}^2 + \dot{y}_{n+1}^2 = 2E/m - 2gz_{n+1}$. It indicates that only two parameters are enough. Here, the more intuitive pare of parameters have been chosen: $(v, d\gamma)$, $v = \sqrt{\dot{x}_{n+1}^2 + \dot{y}_{n+1}^2}$ is the magnitude of the horizontal velocity and $d\gamma$ is the heading angle deflection after one step. Therefore, the model behavior could be captured by a forward map function:

$$(v, \Delta\gamma) = f(\dot{x}_n, \theta, \phi) \tag{5.20}$$

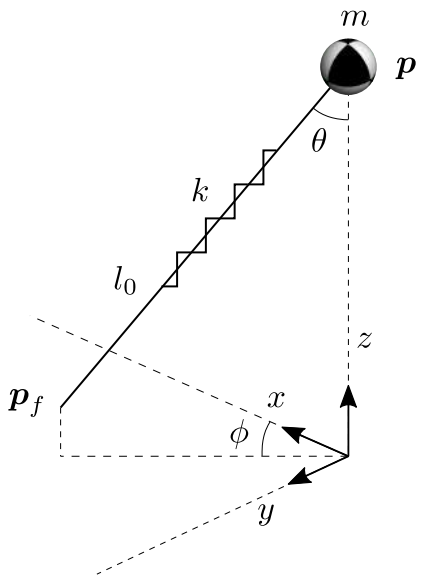


Fig. 5.9 The 3D spring-loaded inverted pendulum (3D-SLIP) model in flight phase.

Given a desired heading velocity v^* and heading deflection $\Delta\gamma^*$ with respect to current apex state, the reverse relation ship emerges:

$$(\theta^*, \phi^*) = f^{-1}(\dot{x}_n, v^*, \Delta\gamma^*) \tag{5.21}$$

At this point, the previously proposed neural-network based method could be applied to encode this inverse relationship. With a simulation initialized with forward velocity $\dot{x}_0$, z could be calculated from chosen constant energy level E. The robot enter flight phase with $\dot{x}_0$ at height z. Given a pair of leg angle (θ, ϕ), the landing point is deterministic and can be calculated (l_0 is known). Right after touchdown, the robot enters stance phase, since leg stiffness k is also fixed, the whole stance phase could be simulated until take off even. The following flight phase also needed to be simulated until the apex. At this moment, the terminal velocity is known and $(v, \Delta\gamma)$ can be

decided accordingly. With this whole process of forward simulation, one data sample could be collected as:

$$\begin{cases} \boldsymbol{x} = [\dot{x}_0, v^*, \Delta\gamma^*]^T \\ \boldsymbol{y} = [\theta, \phi]^T \end{cases} \tag{5.22}$$

Repeating this process with various initial conditions and control parameters, the whole data set can be collected. The rest training part is the same as the 2D case described in section 5.2.2. Network structure needs to be updated based on new training data dimension. The same training algorithm could be applied and some training parameters needs to be tuned.

5.6 Conclusion

In this paper we proposed to use a deep neural network to encode the dynamics of a simple template model and then map to the whole-body robot. Different from the non-linear optimization based approach or the classical tabular method, it transfers most of the computations offline. Once trained, the query of learned knowledge is very fast and can be embedded into real-time control framework. The two-dimensional SLIP model long-term dynamics (return map) has been successfully learned by the neural network. The approach itself is general and not limited to this 2D case with low-dimensional inputs and outputs. As explained in previous section, extension to 3D-SLIP model has been briefly added. The next step is to consider the robustness of trained neural network controller. Another aspect would be incorporating more

parameters such as spring stiffness and energy level into the framework which are assumed to be constant in current implementations.

Chapter 6

Summary and Future Work

6.1 Summary

The objective of this dissertation was to explore existing locomotion approaches and contribute to towards more dynamic capabilities for humanoid robots. This work provides a general control framework that could deal with balancing, walking, hopping and running. Based on the required behavior, different template models can be chosen to generate appropriate reference trajectories for feet and CoM. No matter which template model has been used, a finite horizon prediction has to been made to guarantee the long term locomotion stability. An optimization based whole-body controller takes these reference trajectories as input and generates whole-body control commands for humanoid robot at every time step. The combination of a simple-model based long-term predictive planner and a whole-body model based short-term tracking

controller has been proven to be very effective in generating versatile dynamic motions for humanoid robot.

As shown in Chapter 3, without any template model, the whole-body controller alone could deal with many locally stable tasks such as landing on unknown uneven ground, balancing with multiple contacts including hands and feet, riding on two-wheeled self-balancing mobile platform.

In Chapter 4, LIPM regarding several steps of predictive horizon has been considered in the framework enables the robot to perform various types of dynamic walking motion. The existing MPC control scheme has been improved as shown in Chapter 4 to enable the robot to walk without any reference foot placement anchoring. It can be described as a discrete version of "walking without thinking" which only use foot placement to stabilize walking motion without relying on ankle torque to modulate ZMP. As a result, the robot could achieve versatile locomotion modes such as automatic foot placement with single reference velocity command, reactive stepping under large external disturbances, guided walking with small constant external pushing forces, robust walking on unknown uneven terrain, reactive stepping in place when blocked by external barrier. As an extension of this proposed framework, also to increase the push recovery capability of the humanoid robot, two new configurations have been proposed to enable the robot to perform cross-step motions.

In Chapter 5, more dynamic hopping and running motion have been studied. SLIP model has been considered as a better template model for these dynamic motions. This template model also takes long term stability into consideration and provides

footstep placement which allows the robot to stabilize the running cycle as well as modulating the running speed. In this sense, it also falls into the category of MPC scheme. Theoretically, multiple steps predict horizon could be considered for SLIP model. The SLIP model is capable of deadbeat to desired state in one step for a large range of speed, therefore only one step has been considered in our study. Other than traditional model-based analytical approach, a data-driven approach has been proposed to encode the dynamics of the this model. A deep neural network is trained offline with a large amount of simulation data based on the SLIP model to learn its dynamics. The trained network is applied online to generate reference foot placements for the humanoid robot. Simulations have been performed to evaluate the effectiveness of the proposed approach in generating bio-inspired and robust running motions. The method proposed based on 2D SLIP model can be generalized to 3D SLIP model and the extension has been briefly mentioned in section 5.5.

6.2 Future Work

A framework has been build integrating simple template model based planner and whole-body model based controller. A wide range of tasks has been achieved with this framework: balancing in place, multi-contacts, riding transportation tool, walking and running. A number of potential extensions and improvements to the work are listed here.

- Transition between template models. In this work, different template models have been used to generate walking and running motion. The transition between two template models could be useful when the robot wants to switch from one type of motion to the other. Alternatively, some other works combining templates models [47] [165] shows promising results and this could be another approach to have consistent control scheme.

- Angular momentum in planning. In this work, a simple setpoint (zero) are used in whole-body controller to damp out extra angular momentum. No proper planned trajectory is available at this moment however they could be very important for dynamic motion such as running which involves stance phase and flight phase. In flight phase, the momentum is conserved. During the stance phase, the controller tries its best to damp out the extra angular momentum. It means that, rotational motion is not desirable throughout the whole running process which might not be true. This could be tell from the template model as well, considering only point mass and mass-less leg model indicates the neglecting of angular momentum effects. A template model considering angular momentum could potentially improve movement performance.

- Walking step timing. In current walking MPC control scheme, only fixed step time is considered. The robot is following strict clock ticks switching between different support phases. This limits somehow the push recovery capability of the robot. For example when the robot been pushed left during the left support

phase, since the robot can not cross the right swing foot the left further than the left foot, the best strategies would be putting down the right foot as soon as possible to the closest point to the left foot and than take another large left step. In current framework, this is not enabled due to fixed step time setting. The robot has to wait until the end of the whole swing phase to take further reaction steps.

- Time based and event based control. The walking phases switching (left support and right support) is based on time however this is not the case for the running motion. For running motion, the switching between stance phase and flight phase is based on events (touchdown event and take-off event). A time-event hybrid driven re-planning scheme has been briefly mentioned in section 4.2.8. Merging two template models into the same framework still remain unfinished.

- Neural network controller extension to 3D SLIP model has been briefly mentioned in section 5.5. The 3D SLIP model is intrinsically not self stable, more stability analysis are needed or extensive validation on trained neural-network on test sets is necessary.

6.3 Conclusion

In this thesis, a control framework has been developed to enable humanoid robot achieving various types of motions. Different template models have been used to describe the most important dynamics of the robot for different types of motions.

Model predictive control has been applied to reason about long term stability based on simple models. The whole-body controller will tracking the long term goals during current time step considering all physical constraints. With this scheme, balancing, walking, hopping and running have been achieved. For different types of motion, controllers are developed rather separately and seamless switching between motions is not an easy task. Although detail implementation differs from each other, they do share some common characteristics: simple models can indeed capture the main characters of dynamic motions (CoM motion and foot placements). Reasoning about long term behavior based on these simple models greatly increase the robustness of the system. Compliance is crucial to achieve stable contact switching. Angular momentum improve the system performance in general. State estimation and feedback is critical for successful re-planning.

References

[1] *DARPA Robotics Challenge*. https://www.darpa.mil/program/darpa-robotics-challenge, 2015.

[2] T. Takenaka, T. Matsumoto, T. Yoshiike, and S. Shirokura, "Real time motion generation and control for biped robot-2 nd report: Running gait pattern generation," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1092–1099, IEEE, 2009.

[3] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot HRP-2," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 2, pp. 1083–1090 Vol.2, apr 2004.

[4] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and Others, "Overview of the torque-controlled humanoid robot TORO,"

[5] *Atlas, Boston Dynamics*. https://www.bostondynamics.com/atlas, 2018.

[6] T. McGeer and Others, "Passive dynamic walking," *I. J. Robotic Res.*, vol. 9, no. 2, pp. 62–82, 1990.

[7] A. Goswami, B. Espiau, A. Keramane, A. Goswami, Ambarish and Espiau, Bernard and Keramane, A. Goswami, B. Espiau, A. Keramane, and A. Goswami, Ambarish and Espiau, Bernard and Keramane, "Limit cycles and their stability in a passive bipedal gait," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1, pp. 246–251, IEEE, 1996.

[8] S. H. Collins, M. Wisse, and A. Ruina, "A three-dimensional passive-dynamic walking robot with two legs and knees," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 607–615, 2001.

[9] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient Bipedal Robots Based on Passive-Dynamic Walkers," *Science*, vol. 307, pp. 1082–1085, feb 2005.

[10] M. Wisse and J. Van Frankenhuyzen, "Design and construction of MIKE; a 2-D autonomous biped based on passive dynamic walking," in *Adaptive motion of animals and machines*, pp. 143–154, Springer, 2006.

[11] R. Tedrake, T. W. Zhang, M.-f. Fong, and H. S. Seung, "Actuating a simple 3D passive dynamic walker," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5, pp. 4656–4661, IEEE, 2004.

[12] P. A. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. G. D. Karssen, C. Paul, and A. Ruina, "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge," *The International Journal of Robotics Research*, vol. 33, no. 10, pp. 1305–1321, 2014.

[13] J. W. Westervelt, Eric R and Chevallereau, Christine and Choi, Jun Ho and Morris, Benjamin and Grizzle, "Feedback Control of Dyanmic Bipedal Robot Locomotion," 2007.

[14] T. G. McGee and M. W. Spong, "Trajectory planning and control of a novel walking biped," in *Control Applications, 2001.(CCA'01). Proceedings of the 2001 IEEE International Conference on*, pp. 1099–1104, IEEE, 2001.

[15] J. W. J. W. Grizzle, G. Abba, and F. Plestan, "Asymptotically stable walking for biped robots: Analysis via systems with impulse effects," *Automatic Control, IEEE Transactions on*, vol. 46, no. 1, pp. 51–64, 2001.

[16] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," 2003.

[17] M. H. Raibert, H. B. Brown Jr, and M. Chepponis, "Experiments in balance with a 3D one-legged hopping machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[18] J. K. Hodgins and M. H. Raibert, "Biped Gymnastics," *The International Journal of Robotics Research*, vol. 9, no. 2, pp. 115–128, 1990.

[19] R. R. R. Playter and M. Raibert, "Control of a Biped Somersault in 3D," in *IFToMM-jc International Symposium on Theory of Machines and Mechanisms*, vol. 2, pp. 669–674, 1992.

[20] K. Yin, K. Loken, and M. de Panne, "Simbicon: Simple biped locomotion control," in *ACM Transactions on Graphics (TOG)*, vol. 26, p. 105, ACM, 2007.

[21] C. K. Chow and D. H. Jacobson, "Studies of human locomotion via optimal programming," *Mathematical Biosciences*, vol. 10, no. 3-4, pp. 239–306, 1971.

[22] J. Denk and G. Schmidt, "Synthesis of a walking primitive database for a humanoid robot using optimal control techniques," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, pp. 319–326, 2001.

[23] T. L. Brown and J. P. Schmiedeler, "Energetic effects of reaction wheel actuation on underactuated biped robot walking," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2576–2581, IEEE, 2014.

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

[25] L. Roussel, C. Canudas-de Wit, and A. Goswami, "Generation of energy optimal complete gait cycles for biped robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 3, pp. 2036–2041, IEEE, 1998.

[26] K. D. Mombaur, R. W. Longman, H. G. Bock, and J. P. Schlöder, "Open-loop stable running," *Robotica*, vol. 23, no. 1, pp. 21–33, 2005.

[27] A. C. Fang and N. S. Pollard, "Efficient synthesis of physically valid human motion," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 417–426, 2003.

[28] G. Bessonnet, P. Seguin, and P. Sardain, "A parametric optimization approach to walking pattern synthesis," *The International Journal of Robotics Research*, vol. 24, no. 7, pp. 523–536, 2005.

[29] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[30] R. J. Full and D. E. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *The Journal of Experimental Biology*, vol. 202, pp. 3325–3332, 1999.

[31] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pp. 1405–1411, IEEE, 1991.

[32] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 1, no. 4, pp. 239–246, 2001.

[33] M. VUKOBRATOVIĆ and B. BOROVAC, "Zero-Moment Point — Thirty Five Years of Its Life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004.

[34] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1620–1626, IEEE, 2003.

[35] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *IEEE-RAS international conference on humanoid robots*, 2006.

[36] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online Walking Motion Generation with Automatic Foot Step Placement To cite this version : Online Walking Motion Generation with Automatic Foot Step Placement," *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-Decem, no. Ld, pp. 153–158, 2015.

[37] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS international conference on humanoid robots*, pp. 200–207, IEEE, 2006.

[38] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[39] J. Pratt, T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, "Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1117–1133, 2012.

[40] R. Blickhan and R. J. Full, "Similarity in multilegged locomotion: Bouncing like a monopode," *Journal of Comparative Physiology A*, vol. 173, no. 5, pp. 509–517, 1993.

[41] M. Hutter, C. D. Remy, M. A. Höpflinger, and R. Siegwart, "SLIP running with an articulated robotic leg," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, no. May 2014, pp. 4934–4939, 2010.

[42] A. Wu and H. Geyer, "The 3-D Spring-Mass Model Reveals a Time-Based Deadbeat Control for Highly Robust Running and Steering in Uncertain Environments.," *IEEE Trans. Robotics*, vol. 29, no. 5, pp. 1114–1124, 2013.

[43] W. C. Martin, A. Wu, and H. Geyer, "Robust spring mass model running for a physical bipedal robot.," in *ICRA*, pp. 6307–6312, 2015.

[44] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3D-SLIP model," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 5134–5140, IEEE, 2013.

[45] P. M. Wensing and D. E. Orin, "Development of high-span running long jumps for humanoids," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2014.

[46] S. Rezazadeh and J. W. Hurst, "Toward step-by-step synthesis of stable gaits for underactuated compliant legged robots," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 4532–4538, IEEE, 2015.

[47] I. Mordatch, M. De Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 71, 2010.

[48] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pp. 238–244, IEEE, 2005.

[49] M. de Lasa, I. Mordatch, and A. Hertzmann, "Feature-based locomotion controllers," *ACM Transactions on Graphics*, vol. 29, no. 4, p. 1, 2010.

[50] J. Pratt, C.-M. M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001.

[51] B. J. Stephens and C. G. Atkeson, "Dynamic balance force control for compliant humanoid robots," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 1248–1255, IEEE, 2010.

[52] J. Park and O. Khatib, "Contact consistent control framework for humanoid robots," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1963–1969, IEEE, 2006.

[53] L. Sentis, "Synthesis and control of whole-body behaviors in humanoid systems," *Constraints*, no. July, pp. 1–208, 2007.

[54] Y. Abe, M. Da Silva, and J. Popović, "Multiobjective control with frictional contacts," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 249–258, Eurographics Association, 2007.

[55] A. Macchietto, V. Zordan, and C. R. Shelton, "Momentum control for balance," in *ACM Transactions on graphics (TOG)*, vol. 28, p. 80, ACM, 2009.

[56] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1283–1290, 2011.

[57] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Soueres, and J. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *Robotics, IEEE Transactions on*, vol. 29, no. 2, pp. 346–362, 2013.

[58] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 981–988, IEEE, 2014.

[59] P. Wensing and D. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization Hierarchical QP," *Robotics and Automation (ICRA), 2013 IEEE*, 2013.

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

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

[62] A. Hofmann, M. Popovic, and H. Herr, "Exploiting angular momentum to enhance bipedal center-of-mass control," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 4423–4429, IEEE, 2009.

[63] S. Feng, C. G. Atkeson, H. Geyer, K. Sreenath, and J. Pratt, "Online Hierarchical Optimization for Humanoid Control," 2016.

[64] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[65] R. Featherstone, *Spatial_v2*. http://royfeatherstone.org/spatial/v2, 2012.

[66] M. L. Felis, *Rigid Body Dynamics Library*. https://rbdl.bitbucket.io/.

[67] N. G. Tsagarakis, S. Morfey, G. Medrano Cerda, Z. Li, and D. G. Caldwell, "Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, no. October 2014, pp. 673–678, IEEE, 2013.

[68] N. G. Tsagarakis, D. G. Caldwell, F. Negrello, W. Choi, L. Baccelliere, V. G. Loc, J. Noorden, L. Muratore, A. Margan, A. Cardellino, and Others, "WALK-MAN: A High-Performance Humanoid Platform for Realistic Environments," *Journal of Field Robotics*, vol. 34, no. 7, pp. 1225–1259, 2017.

[69] C. Zhou and N. Tsagarakis, "On the Comprehensive Kinematics Analysis of a Humanoid Parallel Ankle Mechanism," *Journal of Mechanisms and Robotics*, vol. 10, no. 5, p. 51015, 2018.

[70] P.-B. Wieber, "Holonomy and nonholonomy in the dynamics of articulated motion," in *Fast motions in biomechanics and robotics*, pp. 411–425, Springer, 2006.

[71] D. E. Orin and A. Goswami, "Centroidal momentum matrix of a humanoid robot: Structure and properties," *dynamics*, vol. 4, p. 6, 2008.

[72] P. M. Wensing, *Optimization and Control of Dynamic Humanoid Running and Jumping*. PhD thesis, The Ohio State University, 2014.

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

[74] R. S. Sutton, A. G. Barto, F. Bach, and Others, *Reinforcement learning: An introduction*. MIT press, 1998.

[75] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, pp. 1–4, 2016.

[76] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *International Conference on Intelligent Robots and Systems*, pp. 2149–2154, 2004.

[77] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository*, 2016.

[78] Russell Smith, *Open Dynamics Engine*. http://ode.org/.

[79] Erwin Coumans, *Bullet Real-Time Physics Simulation*. https://pybullet.org/.

[80] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[81] J. Hsu, N. Koenig, and C. Dave, *http://wiki.ros.org/gazebo_ros_pkgs*.

[82] L. Muratore, A. Laurenzi, E. M. Hoffman, A. Rocchi, D. G. Caldwell, and N. G. Tsagarakis, "Xbotcore: A real-time cross-robot software platform," in *Robotic Computing (IRC), IEEE International Conference on*, pp. 77–80, IEEE, 2017.

[83] J. Park and O. Khatib, "Contact consistent control framework for humanoid robots," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, no. May, pp. 1963–1969, 2006.

[84] L. Sentis, J. Park, and O. Khatib, "Compliant control of multicontact and center-of-mass behaviors in humanoid robots," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 483–501, 2010.

[85] M. De Lasa, I. Mordatch, and A. Hertzmann, "Feature-based locomotion controllers," in *ACM Transactions on Graphics (TOG)*, vol. 29, p. 131, ACM, 2010.

[86] P. M. Wensing, G. Bin Hammam, B. Dariush, and D. E. Orin, "Optimizing foot centers of pressure through force distribution in a humanoid robot," *International Journal of Humanoid Robotics*, vol. 10, no. 03, p. 1350027, 2013.

[87] P. M. Wensing and D. E. Orin, "Improved Computation of the Humanoid Centroidal Dynamics and Application for Whole-Body Control," 2016.

[88] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*, vol. 29. CRC press, 1994.

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

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

[91] B. Stephens, "Humanoid Push Recovery," *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pp. 589–595, 2007.

[92] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[93] H. G. Nguyen, J. Morrell, K. D. Mullens, A. B. Burmeister, S. Miles, N. Farrington, K. M. Thomas, and D. W. Gage, "Segway robotic mobility platform," in *Optics East*, pp. 207–220, International Society for Optics and Photonics, 2004.

[94] F. Grasser, A. D'arrigo, S. Colombi, and A. C. Rufer, "JOE: a mobile, inverted pendulum," *IEEE Transactions on industrial electronics*, vol. 49, no. 1, pp. 107–114, 2002.

[95] X. Ruan and J. Cai, "Fuzzy backstepping controllers for two-wheeled self-balancing robot," in *Informatics in Control, Automation and Robotics, 2009. CAR'09. International Asia Conference on*, pp. 166–169, IEEE, 2009.

[96] C.-C. Tsai, H.-C. Huang, and S.-C. Lin, "Adaptive neural network control of a self-balancing two-wheeled scooter," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 4, pp. 1420–1428, 2010.

[97] J.-X. Xu, Z.-Q. Guo, and T. H. Lee, "Design and implementation of integral sliding-mode control on an underactuated two-wheeled mobile robot," *IEEE Transactions on industrial electronics*, vol. 61, no. 7, pp. 3671–3681, 2014.

[98] R. O. Ambrose, R. T. Savely, S. M. Goza, P. Strawser, M. A. Diftler, I. Spain, and N. Radford, "Mobile manipulation using NASA's robonaut," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2, pp. 2104–2109, IEEE, 2004.

[99] M. A. Diftler, R. O. Ambrose, K. S. Tyree, S. M. Goza, and E. L. Huber, "A mobile autonomous humanoid assistant," in *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, vol. 1, pp. 133–148, IEEE, 2004.

[100] H. Lee, H. J. Choi, J. H. Park, J. H. Lee, and S. Jung, "Center of gravity based control of a humanoid balancing robot for boxing games: BalBOT V," in *ICCAS-SICE, 2009*, pp. 124–128, IEEE, 2009.

[101] S.-H. Hyon, J. G. Hale, and G. Cheng, "Full-body compliant human–humanoid interaction: balancing in the presence of unknown external forces," *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 884–898, 2007.

[102] C. Ott, M. a. Roa, and G. Hirzinger, "Posture and balance control for biped robots based on contact force optimization," *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 26–33, 2011.

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

[104] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[105] C. G. A. Salman Faraji Soha Pouya and A. J. Ijspeert, "Versatile and Robust 3D Walking with a Simulated Humanoid Robot (Atlas) a Model Predictive Control Approach," *IEEE International Conference on Robotics and Automation*, 2014.

[106] S.-H. Lee and A. Goswami, "Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 3157–3162, IEEE, 2010.

[107] M. Hutter, M. A. Hoepflinger, C. Gehring, M. Bloesch, C. D. Remy, and R. Siegwart, "Hybrid operational space control for compliant legged systems," *Robotics*, p. 129, 2013.

[108] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, p. 0278364913519834, 2014.

[109] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[110] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[111] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[112] T. Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a Momentum-Based Control Framework and Application to the Humanoid Robot Atlas," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1650007, 2016.

[113] B. Henze, A. Dietrich, and C. Ott, "An approach to combine balancing with hierarchical whole-body control for legged humanoid robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 700–707, 2016.

[114] Y.-S. Ha and Others, "Trajectory tracking control for navigation of the inverse pendulum type self-contained mobile robot," *Robotics and autonomous systems*, vol. 17, no. 1-2, pp. 65–80, 1996.

[115] S.-C. Lin and C.-C. Tsai, "Development of a self-balancing human transportation vehicle for the teaching of feedback control," *IEEE Transactions on Education*, vol. 52, no. 1, pp. 157–168, 2009.

[116] *http://wiki.ros.org/Robots/RoboSavvy-Balance*. Robosavvy Ltd, 2017.

[117] K. R. Muske and J. B. Rawlings, "Model predictive control with linear models," *AIChE Journal*, 1993.

[118] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online Walking Motion Generation with Automatic Foot Step Placement," *Advanced Robotics*, vol. 24, pp. 719–737, jan 2010.

[119] B. J. Stephens and C. G. Atkeson, "Push recovery by stepping for humanoid robots with force controlled joints," *2010 10th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2010*, pp. 52–59, 2010.

[120] S. Faraji, S. Pouya, C. G. Atkeson, and A. J. Ijspeert, "Versatile and robust 3d walking with a simulated humanoid robot (atlas): A model predictive control approach," in *IEEE International Conference on Robotics and Automation*, pp. 1943–1950, 2014.

[121] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim, "Robust dynamic walking using online foot step optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5373–5378, 2016.

[122] P. B. Wieber, "Viability and predictive control for safe locomotion," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 1103–1108, 2008.

[123] Y. You, S. Xin, C. Zhou, and N. Tsagarakis, "Straight leg walking strategy for torque-controlled humanoid robots," in *2016 IEEE International Conference on Robotics and Biomimetics, ROBIO 2016*, pp. 2014–2019, IEEE, 2016.

[124] I. E. Sutherland and M. K. Ullner, "Footprints in the Asphalt," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 29–36, 1984.

[125] M. H. Raibert, "Trotting, pacing and bounding by a quadruped robot," *Journal of Biomechanics*, vol. 23, no. SUPPL. 1, pp. 83–98, 1990.

[126] P. M. WENSING, G. BIN HAMMAM, B. DARIUSH, and D. E. ORIN, "Optimizing Foot Centers of Pressure Through Force Distribution in a Humanoid Robot," *International Journal of Humanoid Robotics*, vol. 10, no. 03, p. 1350027, 2013.

[127] C. Semini, V. Barasuol, J. Goldsmith, M. Frigerio, M. Focchi, Y. Gao, and D. G. Caldwell, "Design of the Hydraulically Actuated, Torque-Controlled Quadruped Robot HyQ2Max," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 635–646, 2017.

[128] M. Vukobratovic, A. A. Frank, and D. Juricic, "On the stability of biped locomotion," *IEEE Transactions on Biomedical Engineering*, no. 1, pp. 25–36, 1970.

[129] A. Goswami, "Foot rotation indicator (FRI) point: A new gait planning tool to evaluate postural stability of biped robots," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 47–52, 1999.

[130] A. Goswami and W. Kallem, "Rate of change of angular momentum and balance maintenance of biped robots," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, pp. 3785–3790, IEEE, 2004.

[131] T. Yoshikawa, "Manipulability of robotic mechanisms," *The international journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[132] T. Yoshikawa, "Dynamic manipulability of robot manipulators," *Transactions of the Society of Instrument and Control Engineers*, vol. 21, no. 9, pp. 970–975, 1985.

[133] A. Bowling and O. Khatib, "The dynamic capability equations: a new tool for analyzing robotic manipulator performance," *IEEE transactions on robotics*, vol. 21, no. 1, pp. 115–123, 2005.

[134] N. Naksuk and C. S. G. Lee, "Zero moment point manipulability ellipsoid," in *IEEE International Conference on Robotics and Automation*, pp. 1970–1975, 2006.

[135] N. Naksuk and C. S. G. Lee, "Utilization of movement prioritization for whole-body humanoid robot trajectory generation," in *IEEE International Conference on Robotics and Automation*, pp. 1079–1084, 2005.

[136] S. Cotton, P. Fraisse, and A. P. Murray, "On the manipulability of the center of mass of humanoid robots: Application to design," in *ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 1259–1267, American Society of Mechanical Engineers, 2010.

[137] Y. Gu, C. S. G. Lee, and B. Yao, "Feasible center of mass dynamic manipulability of humanoid robots," in *IEEE International Conference on Robotics and Automation*, pp. 5082–5087, 2015.

[138] M. Azad, J. Babič, and M. Mistry, "Dynamic manipulability of the center of mass: A tool to study, analyse and measure physical ability of robots," in *IEEE International Conference on Robotics and Automation*, pp. 3484–3490, 2017.

[139] M. Morisawa, K. Harada, S. Kajita, K. Kaneko, J. Sola, E. Yoshida, N. Mansard, K. Yokoi, and J.-P. Laumond, "Reactive stepping to prevent falling for humanoids.," in *International Conference on Humanoid Robots*, pp. 528–534, 2009.

[140] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, "Online decision of foot placement using singular LQ preview regulation," in *11th IEEE-RAS International Conference on Humanoid Robots*, pp. 13–18, 2011.

[141] J. A. Castano, Z. Li, C. Zhou, N. Tsagarakis, and D. Caldwell, "Dynamic and reactive walking for humanoid robots based on foot placement control," *International Journal of Humanoid Robotics*, vol. 13, no. 02, p. 1550041, 2016.

[142] P. Kryczka, P. Kormushev, N. G. Tsagarakis, and D. G. Caldwell, "Online regeneration of bipedal walking gait pattern optimizing footstep placement and timing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3352–3357, 2015.

[143] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Step timing adjustment: A step toward generating robust gaits," in *IEEE-RAS 16th International Conference on Humanoid Robots*, pp. 35–42, 2016.

[144] W. Hu, I. Chatzinikolaidis, K. Yuan, and Z. Li, "Comparison Study of Nonlinear Optimization of Step Durations and Foot Placement for Dynamic Walking," in *IEEE International Conference on Robotics and Automation*, pp. 433–439, 2018.

[145] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pp. 120–127, 2014.

[146] R. Tedrake, S. Kuindersma, R. Deits, and K. Miura, "A closed-form solution for real-time ZMP gait generation and feedback stabilization," in *IEEE-RAS 15th International Conference on Humanoid Robots*, pp. 936–940, 2015.

[147] M. H. Raibert, "Legged Robots," *Commun. ACM*, vol. 29, no. 6, pp. 499–514, 1986.

[148] R. J. Full and D. E. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, 1999.

[149] R. Alexander and A. S. Jayes, "Vertical movements in walking and running," *Journal of Zoology*, vol. 185, no. 1, pp. 27–40, 1978.

[150] R. Blickhan and R. J. Full, "Similarity in multilegged locomotion: bouncing like a monopode," *Journal of Comparative Physiology A*, vol. 173, no. 5, pp. 509–517, 1993.

[151] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proceedings of the Royal Society B: Biological Sciences*, vol. 273, no. 1603, pp. 2861–2867, 2006.

[152] M. Ahmadi and M. Buehler, "Controlled passive dynamic running experiments with the ARL-monopod II," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 974–986, 2006.

[153] U. Saranli, M. Buehler, and D. E. Koditschek, "RHex: A simple and highly mobile hexapod robot," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.

[154] R. Playter, M. Buehler, and M. Raibert, "BigDog," in *Pro. of SPIE 6230, Unmanned Systems Technology VIII*, pp. 62302O—-62302O, 2006.

[155] J. A. Grimes and J. W. Hurst, "The design of ATRIAS 1.0 a unique monoped, hopping robot," in *Proceedings of the 2012 International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pp. 548–554, 2012.

[156] W. J. Schwind, *Spring loaded inverted pendulum running: a plant model.* PhD thesis, Electrical Enginnering: Systems, University of Michigan, 1998.

[157] S. G. Carver, N. J. Cowan, and J. M. Guckenheimer, "Lateral stability of the spring-mass hopper suggests a two-step control strategy for running," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 19, no. 2, p. 26106, 2009.

[158] M. H. Raibert and F. C. F. C. Wimberly, "Tabular control of balance in a dynamic legged system," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-14, no. 2, pp. 334–339, 1984.

[159] D. Koepl and J. Hurst, "Force control for planar spring-mass running," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3758–3763, IEEE, 2011.

[160] H. Herr, A. Seyfarth, and H. Geyer, "Speed-adaptive control scheme for legged running robots," 2007.

[161] M. Hutter, C. D. Remy, M. A. Höpflinger, and R. Siegwart, "Slip running with an articulated robotic leg," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 4934–4939, IEEE, 2010.

[162] Y. You, Z. Li, D. G. Caldwell, and N. G. Tsagarakis, "From one-legged hopping to bipedal running and walking: A unified foot placement control based on regression analysis," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 4492–4497, IEEE, 2015.

[163] N. G. Tsagarakis, D. G. Caldwell, A. Bicchi, F. Negrello, M. Garabini, W. Choi, L. Baccelliere, V. Loc, J. Noorden, M. Catalano, and Others, "WALK-MAN: A High Performance Humanoid Platform for Realistic Environments," *Journal of Field Robotics (JFR)*, 2016.

[164] A. De and D. E. Koditschek, "Averaged anchoring of decoupled templates in a tail-energized monoped," in *Robotics Research*, pp. 269–285, Springer, 2018.

[165] T. Apgar, P. Clary, K. Green, A. Fern, and J. Hurst, "Fast Online Trajectory Optimization for the Bipedal Robot Cassie," *Robotics: Science and Systems*, vol. XIV, 2018.