Fakultät für Maschinenwesen
Lehrstuhl für Angewandte Mechanik

Simulation and Control of Biped Walking Robots

Dipl.-Ing. Univ. Thomas Buschmann

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. rer. nat. Ulrich Walter

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. habil. Heinz Ulbrich
2. Univ.-Prof. Dr.-Ing. habil. Boris Lohmann

Die Dissertation wurde am 9. September 2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 18. November 2010 angenommen.

Author

Thomas Buschmann
Lehrstuhl für Angewandte Mechanik
Technische Universität München
85747 Garching
Germany
E-Mail: ThesisThomasBuschmann@googlemail.com

A printed version is available from Verlag Dr. Hut, München (ISBN 978-3-86853-804-5)
An electronic version is available at http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20101201-997204-1-6.

Abstract

This PhD thesis covers simulation and control of biped walking robots. Simulations and experiments were performed with the robots Johnnie and Lola developed at the Institute of Applied Mechanics, Technische Universität München, Germany.

The chapter on modeling and simulation describes the approach to simulating the rigid multibody dynamics. It includes specialized models for simulating drive friction, nonlinear drive kinematics and the unilateral ground contact. Component models, contact and differential equation solvers are combined to a family of robot simulations with varying modeling depth and computational cost.

The thesis also presents a hierarchical system for real-time walking control. Novel aspects include a trajectory generator based on spline collocation and a stabilizing controller based on hybrid position/force control. Lola's redundant toe and pelvis joints are used to reduce joint speeds and avoid joint limits. All methods were verified on both robots in walking experiments and simulations. In experiments Lola reached a maximum walking speed of 3.34 km/h. Using a computer vision system developed at the Institute of Autonomous Systems Technology at the Universität der Bundeswehr in Munich, Germany, Lola is also capable of safely exploring previously unknown environments.

Acknowledgments

First, I would like to thank my supervisor Professor Heinz Ulbrich for giving me the opportunity to do this research as well as for his guidance and support throughout the past six years. He gave me the freedom to pursue my own ideas and created an excellent research environment at the Institute of Applied Mechanics. I feel grateful for having had the opportunity to work on such an interesting and challenging project. I would also like to acknowledge Professor Boris Lohmann and Professor Ulrich Walter for serving on my thesis defense committee. This thesis would not have been possible without the pioneering work on legged locomotion at the Institute of Applied Mechanics led by Professor Friedrich Pfeiffer. His advice and guidance have been invaluable.

I am deeply grateful for having had the chance to work with a number of very talented and highly motivated people. I am especially thankful to Sebastian Lohmeier, who was responsible for Lola's mechatronic system architecture and who did the detailed mechanical design (see [62]). The collaboration with him was most enjoyable, inspiring and productive. I would like to thank Valerio Favot, who worked on Lola's decentral controllers and communication system, and Markus Schwienbacher, who worked on Lola's hardware and software. Without their terrific work, Lola would not have taken a single step. I owe special thanks to Georg Mayr. His long experience with legged robots, his work on Lola's electronics and his help in maintaining the robot Johnnie were invaluable. I would also like to thank Mathias Bachmayer for designing Lola's DSP boards. Experimental robotics research is impossible without decent hardware and I am especially grateful for Wilhelm Miller, Walter Wöß, Simon Gerer, Philip Schneider and Tobias Schmidt's excellent work in manufacturing Lola.

Special thanks are also due to Michael Gienger and Klaus Löffler for their work in developing the robot Johnnie that served as a research platform for many years until Lola was fully operational. Their work was both an inspiration and the basis for the development of Lola.

I would also like to express my gratitude to Gerhard Rohe and Felix von Hundelshausen from the Universität der Bundeswehr in Munich, Germany, who developed Lola's computer vision system. Working with both was an extremely pleasant and rewarding experience and this thesis would have been incomplete without the research on autonomous locomotion.

I am very grateful to Alexander Ewald for organizing the presentation of Lola at the *Hannover Messe* and to Dr. Thomas Thümmel for his support and encouragement throughout this project.

I would also like to thank my (ex-) colleagues Sebasitan Lohmeier, Markus Schwienbacher and Gerhard Schillhuber and my sister Kathrin for proofreading this thesis.

Finally, I would like to acknowledge the DFG's financial support for this research through the research program "Natur und Technik intelligenten Laufens" (Nature and technology of intelligent walking, DFG grant UL 105/29).

Contents

1 Introduction

During the past three decades research and development in robotics has expanded from traditional industrial robot manipulators to include autonomous and animal-like or humanoid robots. Over the past two decades, the field of humanoid robotics has witnessed significant advances. This development has been driven by improvements in actuator, computer and other enabling technologies and guided by the vision of building machines with (some) human-like capabilities.

A machine with human-like appearance and capabilities would be able to operate in all environments designed for humans, such as factories, offices and homes. Also, interacting with humans using natural language and gestures both simplifies the interaction for the human and decreases the psychological barrier for the use of such machines in service applications [19]. All this makes service robotics one of the most promising application areas for humanoids, while the entertainment industry is also exploring the potential of such machines. Even without high-level intelligence, biped robots are potentially superior to wheeled or tracked vehicles in complex and cluttered environments, since they can climb stairs or step onto or over large obstacles.

Building truly humanoid robots will require significant advances in areas including, among others, high-level cognition, computer vision, speech synthesis, speech recognition, manipulation and biped locomotion. Recent interest in humanoid robotics has spawned a large number of research projects focusing either on individual problems or on systems integration. This thesis is the result of work conducted during the research project "Ein leistungsgesteigerter, autonomer Zweibeiner" (An autonomous biped with enhanced performance)[1]. It is part of the DFG[2] package proposal "Natur und Technik intelligenten Laufens" (Nature and technology of intelligent walking)[3], which focuses on biological and technological aspects of autonomous and legged locomotion.

1.1 Problem Statement

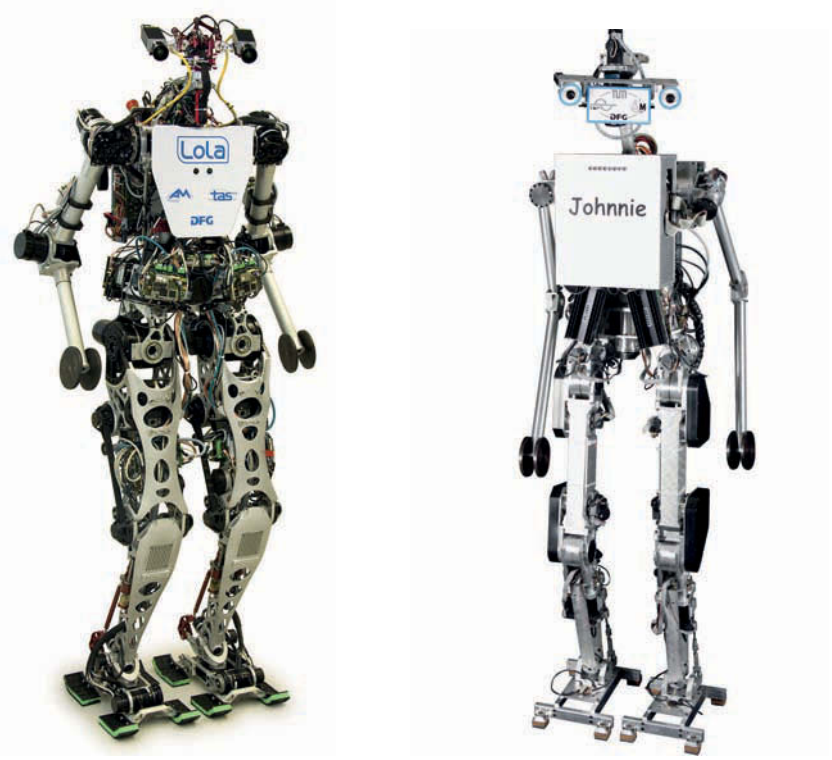
This thesis concerns the modeling, simulation and real-time control of a specific kind of biped walking robot, such as Lola and Johnnie developed at the Institute of Applied Mechanics, Technische Universität München. Figure 1.1 shows photographs of both robots as well as some basic technical data.[4] The type of robot discussed

1 DFG grant UL 105/29

2 The DFG is the "central, self-governing research funding organization in Germany." (http://www.dfg.de)

3 DFG grant PAK 146, http://www.uni-koeln.de/math-nat-fak/zoologie/tierphysiologie/dfgGruppe.html

4 In addition to the 14 joints mentioned in Figure 1.1, Johnnie has two shoulder adduction joints and one for rotating the upper body. However, they were never operational and are



Height	1.8 m
Weight	63 kg
Driven joints	25

Lola

Height	1.75 m
Weight	50 kg
Driven joints	14

Johnnie

Figure 1.1: Biped robots Lola (left) and Johnnie [4] (right)

here is actuated by electric motors and reduction gears and has position, but no force sensing in the joints. It is equipped with force/torque sensors (FTS) in the feet, an inertial measurement unit (IMU) mounted on the pelvis or torso and a stereo camera system used for high-level navigation and control. The robot structure is quite rigid, while elasticity and damping are intentionally introduced in the foot-ground contact. Other examples of such devices include Honda's robots, the HRP-robots and Hubo (cf. Section 1.2.2). Figure 1.2 illustrates the basic structure of this type of robot.

Biped walking is a process of alternatingly supporting the body's weight with one leg while moving the other leg to a new foothold. During this activity, forces generated by gravity and accelerated limbs must be balanced by contact forces acting on the supporting foot or feet. Feasible contact forces are limited due to the unilateral foot-ground contact. This strongly limits the range of physically feasible motions and complicates the process of maintaining balance. In walking pattern generation

[4] therefore not considered in this thesis. Also, the camera head shown on Johnnie in Figure 1.1 has two driven joints. The head was developed at a different institute and its control was not integrated into the robot's walking control. Instead, it was directly controlled by a vision processing system developed by Schmidt et al. [16]. Since the head was not used for work presented here, its joints are not considered in this thesis.

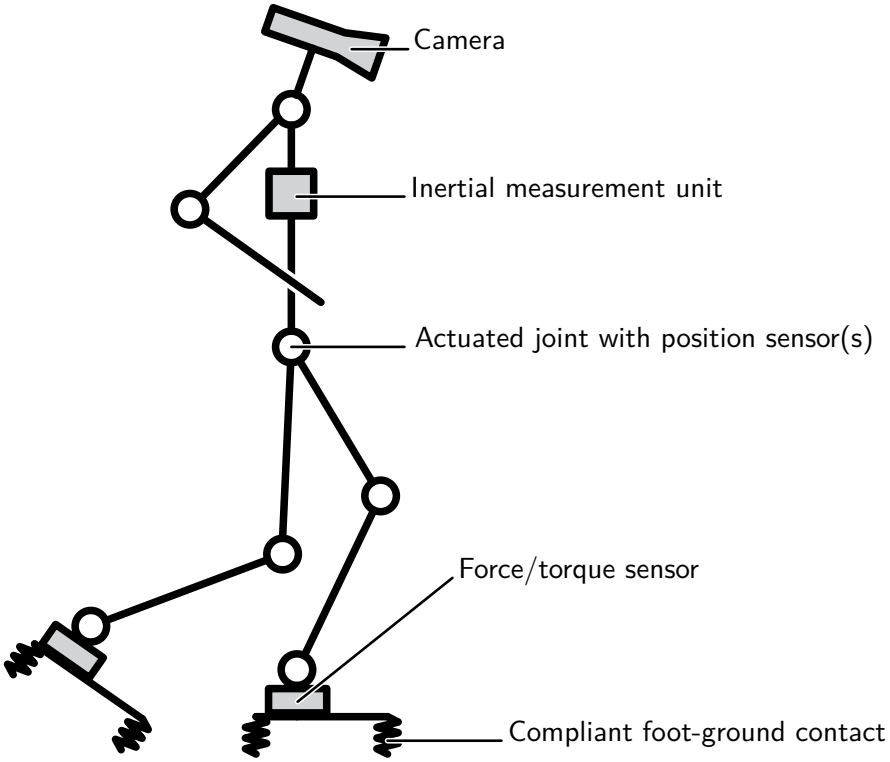


Figure 1.2: Basic structure of biped robots considered in this thesis

and stabilizing control, *feasibility* conditions take the form of inequality constraints. Violating these constraints is physically impossible. However, if the boundary of the feasible region is reached, the contact state will change, i.e., the robot will start tipping over. While controlling individual joints is straightforward, this is not equivalent to *maintaining balance*, since the compliant foot-ground contact makes the robot *underactuated*, i.e., the system has more degrees of freedom (DoFs) than inputs. *Real-time control* is especially challenging, since the complex dynamics of the robot must be approximated to reduce the computational cost and enable real-time execution of planning and control algorithms. At the same time, relevant physical effects must be taken into account in order to achieve fast and robust walking.

To enable autonomous locomotion, the robot must identify viable footstep locations. In cluttered environments this means identifying a collision-free path from the current to the target position. This can be achieved by combining the robot's walking control with a computer vision system capable of recognizing obstacles and choosing a viable path. This thesis does not concern aspects of computer vision and path planning. However, to enable *autonomous locomotion*, the real-time control system must supply information about the robot's state to the vision system and must be able to execute continuously changing commands coming from the path planning module.

Systematic design of walking control algorithms is greatly simplified by *dynamics simulations*, since they allow a detailed analysis of dynamics, stability issues, the influence of sensor noise, etc. Dynamics simulations also are the basis for calculating loads that are required for hardware design. Especially during the early stages of hardware design a large number of different kinematic configurations, motors, gears and other parameters must be evaluated [62]. Therefore, a library of robot models of

varying complexity is an essential tool for analysis, hardware design and model-based real-time control.

1.2 Background and Related Work

1.2.1 A Short History of Humanoid and Animal-Like Robots

Cultural History of Robots

There is a long history of building animal and human-like machines and an even longer one of references to such machines in literature. Often, human-like machines or machine-like humans are used as metaphors for the dangers of technology and human hubris. Well-known examples include the *Golem* from the Bible, which is originally described as a perfect servant, whose only fault is being too literal in fulfilling his master's orders, but later becomes a raging monster in the German Expressionist film "The Golem" by Paul Boese from 1920.

In "*Frankenstein*: or, the Modern Prometheus" by Mary Wollstonecraft Shelly, the Swiss scientist Victor Frankenstein creates an artificial man from pieces of corpses. The artificial being seeks affection, but is rejected by everyone and eventually Victor Frankenstein dies.

In Karel Čapek's play "R.U.R.: Rossum's Universal Robots" (published 1920, performed 1921), a scientist succeeds in building human-like machines superior to humans in both precision and reliability. The machines start to dominate the human race and threaten it with extinction, but humanity is saved at the last moment. The word "robot," which is derived from the Czech word for forced labor, has its origin in this play.

Only in more recent science fiction novels robots are literally meant to be human-like machines and therefore do not necessarily have to be evil or dangerous. A prime example are Isaak Asimov's popular science fiction novels.

Mostly due to this history, significant parts of the general public perceive humanoid robots either as a threat to humans or have unrealistic expectations about the abilities of such machines. This cultural background is important, since the general public must accept and have realistic expectation about the abilities of humanoids if they are to become widely used. Accordingly, beyond the obvious technological hurdles that must be overcome, the cultural history of humanoid robots poses additional challenges to the widespread use of such devices

Technological History of Robots

In the past, there were several distinct phases in which scientist and engineers tried to build machines with human or animal-like capabilities. Each phase corresponds to significant advances in technology. Building such machines has been motivated both by the hope of actually replicating human or animal-like behavior and to showcase what is possible with cutting edge technology.

Early examples of purely mechanical, animal-like mechanisms were built in Greece by Archytas of Tarantum in 350 BC. Automatons such as a mechanical orchestra were also popular in imperial circles in China from the third century BC until the 13th century AD.

During the 1800s, significant advances in mechanical engineering lead to the creation of many automatons in western Europe. These automatons were complex, mainly mechanical devices that used important technological advances that also lead to the invention of the (commercial) steam engine and the automated loom. Interestingly, some of the most sophisticated mechanisms were designed by Jaques de Vaucason (1709-1782), who also invented the first automated loom, which was driven by perforated cards.

During the 1940s, advances in electronics, drives and mass production enabled the design of much more sophisticated electromechanical robots using technologies also found in cars and consumer goods of those days. One example is the robot Electro, developed by Westinghouse and presented at the 1939 World's Fair in New York, which was driven by electric motors and controlled by vacuum tube relays. The robot was unable to walk, but could move using small wheels attached to the feet, smoke and emulate a conversation using a record player. Although primitive by today's standards, the development of Electro cost Westinghouse several hundred thousand dollars [68].

The development of modern humanoid robots started in the late 1960s, driven mainly by advances in sensors, actuators and computer technology that also enabled sophisticated factory automation and complex consumer electronics.

1.2.2 Related Work

There is a large and growing body of research on humanoid and legged robots. Due to the very large number of publications in this field, only the work considered to be the most relevant for this thesis is summarized briefly in the following. In particular, only work on fully actuated, three dimensional biped robots is considered. Surveys of legged walking and running robots can be found in [60, 97, 99, 110]. Gienger [25] and Lohmeier [62] give detailed surveys of biped robot hardware design. More detailed reviews of specific aspects of simulation and control are given in the chapters covering these subjects.

Waseda University

In 1973, Kato initiated modern research on biped walking robots at the Waseda University by developing Wabot-1, the first full-scale human-like robot. It was hydraulically powered and used a static gait. A number of advanced robots were subsequently developed at Waseda, the latest one is Wabian-2 [79]. Its basic hardware structure corresponds to that shown in Figure 1.2. The stability and performance of this system has been proven in many experiments. It has been used for testing innovative hardware concepts including passive toes and feet with a human-like arch structure [117].

Tokyo University

In the 1980s, Miura and Shimoyama developed several very small biped robots, notably the robot Biper-3 [64] with point feet. The robot is modeled with linearized equations of motion (EoM) and the double support period is assumed to be instantaneous. The robot is stabilized by adjusting the stride length based on a limit cycle analysis of a simplified linear robot model.

More recently, the humanoid robots H6 and H7 [77] were designed and built by Inoue at Tokyo University's JSK-Laboratory in cooperation with Kawada Industries. These robots were equipped with active toe joints, enabling them to walk faster and climb higher stairs than would have been possible with one-segment feet. The work on vision-guided walking [76] and whole-body motion control [78] is especially notable.

MIT/Carnegie Mellon Leg Laboratory

Raibert developed several hopping machines in the 1980s with one, two or four legs [34, 97]. These robots cannot walk, but are able to hop and run very fast and stably. The robots have telescopic legs controlled by pneumatic and hydraulic actuators. An umbilical cable transfers sensor data to an external computer and provides electrical, hydraulic and pneumatic power. This design allows very lightweight legs that account for only approximately 5% of the robot's mass. This enables very simple, yet effective balancing control based on stabilizing the limit cycle by adjusting the stride length. Although these concepts have yet to be applied to full-sized humanoid robots able to perform general tasks other than hopping or walking, this ground breaking work has been very influential.

Boston Dynamics

Boston Dynamics[5] recently presented a biped robot called Petman capable of walking at 7.08 km/h[6]. The project is funded by DARPA (the U.S. Department of Defense's research and development office) and to the author's knowledge no substantial information on the hardware and control system has been published to date. However, the company states that it is based on Big Dog's technology, a quadruped robot also developed by Boston Dynamics. Big Dog is driven by hydraulic actuators and is equipped with an on-board internal combustion engine and pump. Publications on Big Dog's control system give little detail [98]. However, the available videos and publications suggest that the same basic approach taken by Raibert in his work on hopping robots is pursued with Big Dog and Petman: hydraulic actuation with on-board power generation lead to a high body mass/leg mass ratio, simplifying robot dynamics, which in turn simplifies balancing by controlling the limit cycle through adaptation of footstep locations. Both Big Dog and Petman have a three-segmented leg, which avoids the typical "knee singularity" of most biped robots but also leads to a non-humanoid morphology for Petman. Furthermore, the robot does not seem to have two-DoF ankle joints or a hip rotation joint. This reduces the leg inertia and provides enough DoFs for straight walking. However, this makes Petman a specialized machine for straight walking. Performing more general tasks, or even curve walking or walking sideways, requires a minimum of six DoFs in each leg. The video released on the company website shows the potential for this design approach in building very fast robots. However, the design also seems to preclude the indoor use of such a device as a general-purpose service robot.

5 Boston Dynamics was founded as a spin-off from the Massachusetts Institute of Technology by Marc Raibert and others.

6 http://www.bostondynamics.com/robot_petman.html

Honda

Honda achieved significant advances in biped walking technology during a secret research program initiated in 1986. From 1986 to 1993 the prototypes E1 to E6 were developed. The robot E4 reached a top speed of 4.7 km/h [35] using trajectories generated off-line and a simple ankle joint impedance control. The robots P1 to P3, developed from 1993 to 1997, were equipped with arms and grippers and were the first fully self-contained humanoid robots. While the P1 robot was quite large at 190 cm, the size was reduced to 182 cm for P2, 160 cm for P3 and finally 120-130 cm for the Asimo robots.

Honda has published very few scientific papers, but quite a bit can be learned from patents or patent applications (for example [119, 120]). The basic technology of the P2 robot is described in [32]. Recently, some details of trajectory planning and control methods used for the Asimo robots, previously available only as patent applications, were published as scientific papers [121–124].

Honda's approach to building biped robots has been both very influential and successful. Many researchers have adopted the same basic approach of using six-DoF legs, force/torque sensors in the feet, an IMU in the torso and electric motors in combination with harmonic drive gears. The same can be said for trajectory planning and control, where using a lumped mass model and controlling upper body orientation through contact moments has been widely adopted. At 10 km/h, Honda currently has the fastest running biped robot [122].

Humanoid Robotics Project / AIST

The Japanese National Institute of Advanced Industrial Science and Technology (AIST) research center developed a number of sophisticated humanoid robots together with Kawada Industries and a number of Japanese universities. Development started with the "humanoid robotics project" (1998 to 2002) funded by the Japanese Ministry of Economy, Trade and Industry (METI). The project started with a P3 robot provided by Honda which was called HRP-1. The size and basic design approach is similar for HRP-2 [48] and HRP-3 [49]. The newest robot HRP-4 is based on the same technology, but is distinguished by a very slender shape, a female appearance and an 8-DoF face [47]. HRP-2's maximum speed is 2.5 km/h [48], while HRP-3's is 2.0 km/h [51].

Toyota

Toyota has presented several "partner robots" that can perform a number of tasks such as walking or playing musical instruments [125]. A more notable development is Toyota's running robot [116]. At 7 km/h, this briefly was the world's fastest biped robot. It is distinguished by having active toe joints and no force/torque sensors. Balancing is achieved by estimating the center of gravity (CoG) position and replanning CoG trajectories and footstep locations. The very promising approach of fast replanning of (CoG) trajectories using sensor feedback was previously proposed by Nishiwaki and demonstrated on a modified HRP-2 robot [74].

KAIST

The Korea Advanced Institute of Science and Technology (KAIST) developed several humanoid robots, the latest one being HUBO-2. At 130 cm it is the same size as

Honda's Asimo. The hardware and control follow the same basic approach taken by Honda and AIST. Using trajectories generated off-line, the robot can walk at 1.4 km/h and run at a speed of 3.24 km/h. For both walking and running the step length is shorter than the foot length, i.e., a statically stable trajectory can be used in walking direction. Balance control is based on several decoupled, linear controllers acting on individual joints [14].

Yobotics/ IHMC

Pratt et al. presented the Yobotics-IHMC (Florida Institute of Human and Machine Cognition) lower body humanoid robot, which was developed by Yobotics, Inc. in cooperation with several universities [93]. It uses "series elastic actuators" [101] to drive the 6-DoF legs and "virtual model control," i.e., technology developed for MIT's Spring Flamingo and M2 robots [94]. While the robot is slower and its movements appear to be less smooth than those of, e.g., Asimo or HRP-2, this work is notable since it deviates from the mainstream path of a position-controlled robot stabilized using IMU and FTS data. Balancing is based on "capture points," which are defined as locations where "the robot can step to in order to bring itself to a complete stop." [95].

Institute of Applied Mechanics, Technische Universität München

Building on experience gained during the development of a six-legged, insect-like robot and an eight-legged pipe crawling robot [89], Gienger and Löffler [25, 60] developed the biped robot Johnnie. It is actuated by brushed DC motors and harmonic drive gears are used in all joints except the ankles, where a double spatial slider-crank mechanism with ballscrews is used. Six-axis force/torque sensors in the feet and an IMU in the upper body are used for stabilizing control. In the system developed by Löffler [60], on-line trajectory generation is based on polynomial curves for the motion of the feet. The CoG-trajectory is planned using hyperbolic functions in the lateral plane and a fifth-order polynomial in the sagittal plane. Balancing is achieved by controlling the upper body orientation via the contact moments. Using this system, a maximum walking speed of 2.4 km/h was achieved. Connected to a computer vision system developed by Schmidt et al. [16], Johnnie was the first biped robot to autonomously navigate among obstacles using only on-board perception and control.

University vs. Corporate Research

It is evident from the foregoing overview that currently almost all fast and sophisticated biped robots are developed by companies (Honda, Toyota, Kawada Ind. and Boston Dynamics) or national research centers (AIST and KAIST). Some information has been published as either scientific papers or patents for most robots, but many details are closely guarded secrets. Boston Dynamics has released almost no substantial information on the control systems for Big Dog and Petman. While AIST has published quite a few papers on its robots, many details of the stabilizing controller have not been released[7].

7 In his keynote presentation at the Dynamic Walking 2008 conference, Kajita said that "So far, our controller is not designed based on solid control theory. Thus we have not yet published the related paper."

1.3 Overview of the Thesis

The first objective of this thesis is the development of a modular simulation system for biped robots. On the one hand, the system should be capable of simulating all effects that are relevant for control and hardware design. On the other hand, it should be possible to quickly simulate global system dynamics, since this is very important during the development of the control system.

The second objective is the development of a walking control system for biped robots capable of achieving fast, flexible, autonomous and human-like walking. In particular, the robot should be capable of walking in arbitrary directions: forward and backwards, sideways and around corners. It should also be able to step over obstacles while coping with unknown disturbances such as slight pushes or uneven ground. Furthermore, the system should be sufficiently general to be applicable to robots with different kinematics and mass distribution, without requiring excessive parameter tuning. The main contributions of this thesis are:

- The development of a modular simulation system providing problem-specific robot models, ranging from simpler ones allowing fast simulation of global system dynamics to complex models that include gear elasticity, finite element based-contact models and drive dynamics.
- New or improved algorithms for real-time walking control, including:
  - Trajectory generation
  - Balance control
  - Contact force control
  - Redundancy resolution
  - Fast walking
  - Autonomous locomotion
- Experimental verification of a control system for biped walking on two different robots (Johnnie and Lola)

All modeling, simulation and control methods presented in this thesis were applied to both robots Johnnie and Lola whenever possible. Successful walking experiments were performed with both robots using the same walking control framework.

Chapter 2 describes the *robot and environment models*, as well as the *simulation system* developed for Johnnie and Lola. Background on the *stability and feasibility in biped walking* is presented in Chapter 3. The system for *real-time trajectory generation* is described in Chapter 4 and the *feedback control* system in Chapter 5. An overview of the computer vision system developed at the Autonomous Systems Technology Institute, Universität der Bundeswehr[8] and other control components required for *autonomous walking* are given in Chapter 6. Chapter 7 gives a brief overview of the software system developed for simulation and real-time control. Selected results of walking experiments, simulations and experiments in autonomous

8 http://www.unibw.de/lrt8/index_html-en

locomotion are summarized in Chapter 8. Chapter 9 concludes the thesis with a summary, a discussion and suggestions for future research.

2 Modeling and Simulation

2.1 Introduction

An important tool for hardware and controller design for humanoid robots is a dynamics simulation for calculating loads and analyzing system dynamics, control system robustness and performance. Kinematic and dynamic models are also needed for model-based walking control. However, both required modeling depth and acceptable computational cost vary depending on the task, e.g., when designing stabilizing algorithms or calculating loads for dimensioning new robot hardware. This makes a library of robot models very useful during development.

This chapter describes a modular system for simulating biped walking robots with compliant foot-ground contacts, such as the robots Johnnie and Lola. It is an improved version of the system the author presented in [7].

The system consists of models for the different components of a biped robot and solvers for the resulting EoM. The model library includes rigid body mechanics, gear friction, gear elasticity, electrical motor dynamics, nonlinear drive kinematics, unilateral viscoelastic contacts and polygonal environment models. By combining these elements, a family of robot simulations with varying modeling depth and computational complexity is obtained.

2.1.1 Related Work

Due to the significance of simulation for the development of humanoid robots and the interesting challenges in simulating such complex machines, there is a large and growing number of publications devoted to this topic. A brief overview is given in the following. In many cases, contacts between foot and ground or between different bodies of the multibody system (MBS) are treated as rigid [3, 13, 72]. The non-smooth dynamics with impacts are then described using measure differential inclusions [1].

Nakaoka et al. [72] implemented a dynamics simulation for humanoid robots using a rigid contact model. Rigid multibody mechanics are solved using an $O(N)$ algorithm and the contact problem is solved using an iterative linear complementarity problem (LCP) solver. The compliance of the rubber bushes in the simulated HRP-2 robot [48] are modeled using virtual joints with attached spring-damper systems. Due to the very small mass of the foot's base plate that contacts the environment, "virtual inertia" is added to the sole to avoid high frequency oscillations and small time steps. The system is implemented in C++ and integrated into the overall simulation and control system via CORBA[1].

1 The "Common Object Request Broker Architecture" is a standard for object oriented middleware released by the Object Management Group (http://www.corba.org/)

Wieber et al. [132] presented a framework for simulating both humanoid robots and biomechanical models of the human body. The system also uses a rigid contact model. The models are generated using the computer algebra program Maple[2] and its automatic code generation and integrated into the open source Matlab[3]-clone Scilab[4].

Yamane [134] presented a system for simulating human figures that includes a parallel $O(\log(N))$ algorithm for rigid multibody dynamics. Rigid contacts are implemented using a special iterative solver that is fast but does not treat the general case of multiple contacts exactly. Compliant contacts are simulated using a penalty method.

2.1.2 Overview

Johnnie and Lola are both made of fairly rigid aluminum segments connected by revolute joints and driven by electric motors. Both robots use harmonic drive gears as speed reducers in most joints and parallel mechanisms with ballscrews or planetary roller screws to actuate the ankle joints. Lola also uses planetary roller screw-based mechanisms to actuate the knee joints.[5] While brush DC motors are used for Johnnie, permanent magnet synchronous motors (PMSM) are used for Lola. The feet of the robots are equipped with viscoelastic contact elements that are deformed during walking and have a strong influence on walking dynamics.

The major dynamical effects on biped locomotion for these robots are:

1. Rigid body mechanics
2. Drive friction
3. Gear elasticity
4. Nonlinear kinematics in ankle and knee joints
5. Unilateral, compliant foot-ground contact
6. Electrical motor dynamics

Items (1) to (4) are combined to a set of second order ODEs describing the multibody dynamics. The contact model adds a set of first order ODEs with complementarity conditions for unilateral contact and coulomb friction. Finally, the electrical motor dynamics add another set of first order ODEs.

A minimal coordinate representation is used for all models in order to reduce the computational cost of simulations and enable code reuse for real-time model-based control. Note that the use of minimal coordinates can lead to a larger number of bodies in the MBS than DoFs for some model types. If gear elasticity is neglected, we have a total of $n_{\text{DoF}} = 6 + n_{\text{joint}}$ DoFs for a robot with n_{joint} joints, whereas the

2 http://www.maplesoft.com/
3 http://www.mathworks.com/
4 http://www.scilab.org/
5 For the sake of simplicity, the term “planetary roller screw” is used in the following, even if the results are valid for Johnnie, where only ballscrews are used.

total number of bodies is $2n_{\text{joint}} + 1$, since one segment and one motor is added for every joint.

In order to simulate the robot, the forward dynamics model is combined with sensor models and the walking controller. The basic control-flow of the resulting dynamics simulation is shown in Figure 2.1. The robot state is determined by integrating a set of ODEs modeling the system dynamics. From this information, sensor data used by the walking controller is calculated. This in turn is fed into the walking controller, which calculates the input for the EoM.

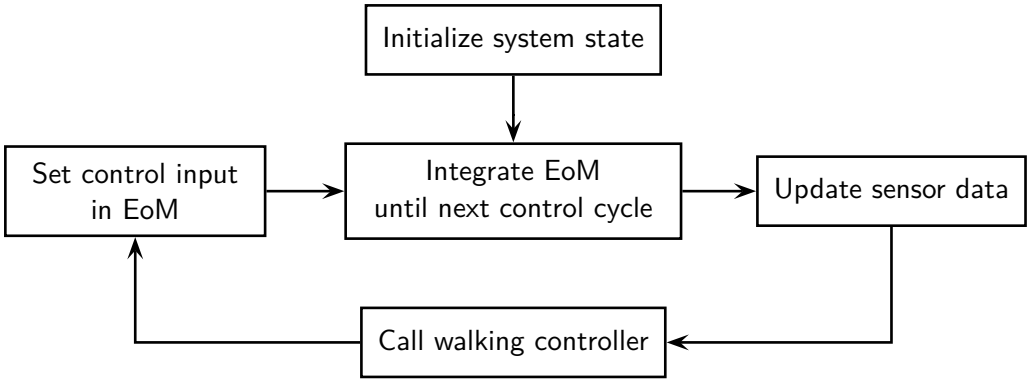


Figure 2.1: Basic structure of the dynamics simulation

2.2 Rigid Body Dynamics

Table 2.1: Kinematics nomenclature

${}^k\boldsymbol{r}_o$	Absolute position of the k-th body's FoR
${}^k\boldsymbol{\omega}$	Angular velocity of k-th body
$\boldsymbol{A}_{ij}$	Rotation matrix transforming vectors from the j-th into the i-th FoR
${}^k\boldsymbol{r}_{pc}$	Position of the k-th body relative to its parent
${}^k\boldsymbol{r}_{\text{CoG}}$	Center of gravity of the k-th body relative to its FoR
$\boldsymbol{J}_R := \partial\boldsymbol{\omega}/\partial\dot{\boldsymbol{q}}$	Rotational Jacobian
$\boldsymbol{J}_{To} := \partial\dot{\boldsymbol{r}}/\partial\dot{\boldsymbol{q}}$	Translational Jacobian
$\boldsymbol{J}_{Rq} := \partial\boldsymbol{\omega}/\partial\boldsymbol{q}$	Rotational Jacobian
$\boldsymbol{J}_{Tqo} := \partial\dot{\boldsymbol{r}}_o/\partial\boldsymbol{q}$	Translational Jacobian

2.2.1 Topology and Degrees of Freedom

The robot's main segments form an open kinematic chain with tree structure. The upper body is chosen as the root link and positions, velocities, Jacobians and other quantities are calculated recursively [7] following a method described, e.g., in [5]. We therefore only need to know the relative kinematics between each body and its parent to calculate the EoM.

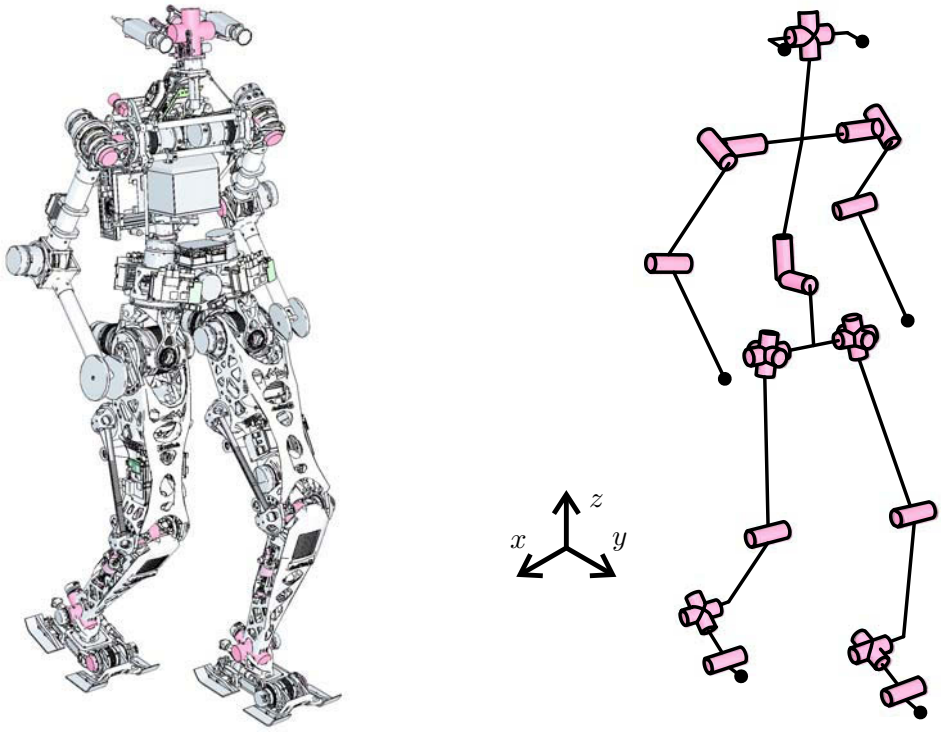


Figure 2.2: Basic kinematic configuration of Lola with joints shown as red cylinders. Coordinate systems not shown for the sake of clarity.

If motor and gear dynamics are taken into account, one body is added to the MBS for each drive. If the gears are assumed to be rigid, this does not add DoFs to the system, since a minimal coordinate representation is used. In the case of elastic gear models, one DoF is added for each drive.

For joints with harmonic drive gears the tree structure is maintained, while the ankle joints are driven by parallel mechanisms that form closed kinematic chains. For Lola a further closed kinematic chain is introduced in the mechanisms driving the knee joints. Formally, the tree structure can be restored by cutting the drive's mechanism open and adding algebraic constraints to the original EoM, which leads to a Differential Algebraic Equation System (DAE). Fortunately, the ankle and knee joint kinematics can be solved analytically, removing algebraic constraints and formally restoring the system's tree structure.

Figure 2.2 shows the kinematic configuration for Lola's major segments. For each body, a local frame of reference (FoR) is introduced. For all bodies except the root link, the origin is located at the joint closest to the root link and the joint axis is chosen as the coordinate system's z-axis. Additionally, an inertial FoR is introduced with the z-axis pointing upwards in the opposite direction as gravity.

The robot's tree structure leads to the local topology shown in Figure 2.4. Due to the tree structure, kinematic quantities for each body can be calculated from those for its parent and the relative kinematics describing the coupling to the parent, i.e., the robot kinematics can be calculated recursively starting at the root link. In the following, the basic approach to calculating the robot's kinematics is presented.

Figure 2.3: Sub- and superscripts for a variable "$\boldsymbol{r}$"

2.2.2 Recursive Kinematics Calculation

Minimal Coordinate Representation

Since the robot is modeled using a minimal coordinate representation, kinematic quantities must be expressed as functions of the generalized coordinates $\boldsymbol{q}$. All the system's constraints are scleronomic and holonomic, which leads to the following equations for the (absolute) velocity $\dot{\boldsymbol{r}}$, acceleration $\ddot{\boldsymbol{r}}$, angular velocity $\boldsymbol{\omega}$ and angular acceleration $\dot{\boldsymbol{\omega}}$ of one body in the MBS [126]:

$$ {}_k\dot{\boldsymbol{r}} = \frac{\partial\, {}_k\dot{\boldsymbol{r}}}{\partial\, \dot{\boldsymbol{q}}}\dot{\boldsymbol{q}} = {}_k\boldsymbol{J}_{To}\,\dot{\boldsymbol{q}} \tag{2.1} $$

$$ {}_k\ddot{\boldsymbol{r}} = \frac{\partial\, {}_k\dot{\boldsymbol{r}}}{\partial\, \dot{\boldsymbol{q}}}\ddot{\boldsymbol{q}} + \frac{\partial\, {}_k\dot{\boldsymbol{r}}}{\partial\, \boldsymbol{q}}\dot{\boldsymbol{q}} + \boldsymbol{\omega} \times {}_k\dot{\boldsymbol{r}} = {}_k\boldsymbol{J}_{To}\,\ddot{\boldsymbol{q}} + {}_k\boldsymbol{J}_{Tqo}\,\dot{\boldsymbol{q}} + {}_k\boldsymbol{\omega} \times {}_k\dot{\boldsymbol{r}} \tag{2.2} $$

$$ {}_k\boldsymbol{\omega} = \frac{\partial\, {}_k\boldsymbol{\omega}}{\partial\, \dot{\boldsymbol{q}}}\dot{\boldsymbol{q}} = {}_k\boldsymbol{J}_{R}\,\dot{\boldsymbol{q}} \tag{2.3} $$

$$ {}_k\dot{\boldsymbol{\omega}} = \frac{\partial\, {}_k\boldsymbol{\omega}}{\partial\, \dot{\boldsymbol{q}}}\ddot{\boldsymbol{q}} + \frac{\partial\, {}_k\boldsymbol{\omega}}{\partial\, \boldsymbol{q}}\dot{\boldsymbol{q}} = {}_k\boldsymbol{J}_{R}\,\ddot{\boldsymbol{q}} + {}_k\boldsymbol{J}_{Rq}\,\dot{\boldsymbol{q}} \tag{2.4} $$

The nomenclature used for kinematic quantities of the MBS is listed in Table 2.1.

Upper Body Kinematics

As root link of the MBS, the upper body has six DoFs. Rotations are described using z-x-z Euler angles $(\psi, \vartheta, \varphi)$. Translations are given by the position of the FoR relative to the inertial FoR. The upper body's DoFs are assigned to indices $i = 0 \ldots 5$ in the vector of generalized coordinates $\boldsymbol{q}$. Therefore, the kinematic quantities for the upper body are simply calculated from $\boldsymbol{q}, \dot{\boldsymbol{q}}$ without further dependencies:

$$ \boldsymbol{A}_{uI} = \boldsymbol{A}_{uI}(\psi, \vartheta, \varphi) \tag{2.5} $$

$$ {}^{u}_{u}\dot{\boldsymbol{r}}_o = \boldsymbol{A}_{uI} \begin{pmatrix} \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \end{pmatrix} \tag{2.6} $$

$$ \boldsymbol{\omega}_u = \frac{\partial\, \boldsymbol{\omega}_u}{\partial\, \dot{\boldsymbol{q}}}\dot{\boldsymbol{q}} \tag{2.7} $$

Here u denotes the upper body reference frame. The representation of spatial orientation used for the upper body is listed in Appendix D. Throughout this thesis, leading subscripts denote the FoR and leading superscripts the reference body, while normal subscripts are used for reference points and indices. The meaning of sub- and superscripts is illustrated in Figure 2.3. In this nomenclature, ${}^{u}_{u}\dot{\boldsymbol{r}}_o$ denotes the velocity of the upper body's origin o described in the upper body's FoR u.

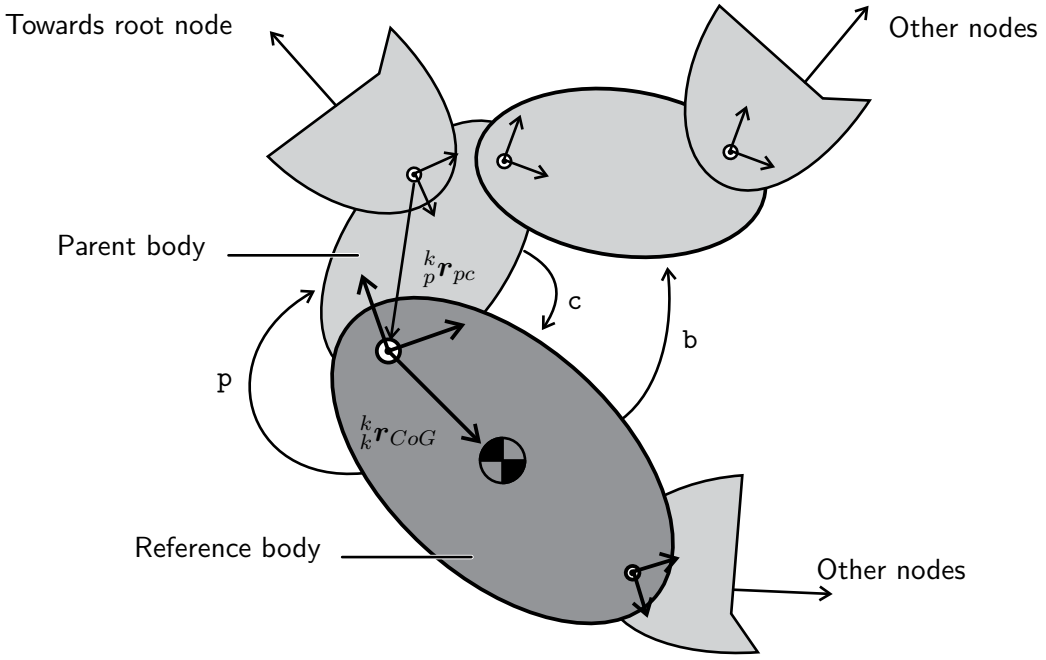


Figure 2.4: Local description of tree structure with pointers to parent (p), child (c) and brother (b) nodes and vectors to the center of gravity ${}^{k}_{k}\boldsymbol{r}_{\mathrm{CoG}}$ and from parent to child node ${}^{k}_{p}\boldsymbol{r}_{pc}$.

Kinematics of Segments and Drives

For all other bodies, kinematic quantities are calculated from those of the parent link and equations describing the coupling to the parent (p). For example, the velocity $\dot{\boldsymbol{r}}$ and angular velocity $\boldsymbol{\omega}$ of the k-th body are given by:

$$ {}^{k}_{k}\dot{\boldsymbol{r}}_{o} = {}^{k}\boldsymbol{A}_{kp}\left({}^{p}_{p}\dot{\boldsymbol{r}}_{o} + {}^{p}_{p}\boldsymbol{\omega} \times {}^{k}_{p}\boldsymbol{r}_{pc}\right) \tag{2.8} $$

$$ {}^{k}_{k}\boldsymbol{\omega} = {}^{k}\boldsymbol{A}_{kp}\,{}^{p}_{p}\boldsymbol{\omega} + {}^{k}_{k}\boldsymbol{\omega}_{rel} \tag{2.9} $$

Translational and rotational Jacobians are obtained by differentiating (2.8) and (2.9) with respect to the generalized velocities $\dot{\boldsymbol{q}}$:

$$ {}^{k}_{k}\boldsymbol{J}_{To} := \frac{\partial\,{}^{k}_{k}\dot{\boldsymbol{r}}_{o}}{\partial\,\dot{\boldsymbol{q}}} = {}^{k}\boldsymbol{A}_{kp}\left({}^{p}_{p}\boldsymbol{J}_{To} + {}^{p}_{p}\boldsymbol{J}_{R} \times {}^{k}_{p}\boldsymbol{r}_{pk}\right) \tag{2.10} $$

$$ {}^{k}_{k}\boldsymbol{J}_{R} := \frac{\partial\,{}^{k}_{k}\boldsymbol{\omega}}{\partial\,\dot{\boldsymbol{q}}} = {}^{k}\boldsymbol{A}_{kp}\,{}^{p}_{p}\boldsymbol{J}_{R} + {}^{k}_{k}\boldsymbol{J}_{Rrel} \tag{2.11} $$

The generalized coordinates $\boldsymbol{q}$ are chosen such that $i \geq j$, if q_i describes the relative motion of a body further down the forward recursion tree than q_j. Therefore, only submatrices have to be calculated, greatly reducing the computational cost.

2.2.3 Relative Kinematics

Segments and Harmonic Drives

For both segments and harmonic drive gears, the relative kinematics with respect to the parent link are described by a 1-DoF rotation about the body's z-axis. Therefore, the relative angular velocity and rotational Jacobian are given by:

$$ {}^{k}_{k}\boldsymbol{\omega}_{rel} = \boldsymbol{e}_z \dot{\varphi}_k = \boldsymbol{e}_z N_k \dot{q}_{i_k} \tag{2.12} $$

$$ {}^{k}_{k}J_{Rrel,[i,j]} = \begin{cases} N_k & \text{for} \quad i = 2 \wedge j = q_k \\ 0 & \text{otherwise} \end{cases} \tag{2.13} $$

Here, N_k is the gear ratio, which is equal to 1 for segments, i_k is the index for the k-th body into the vector of generalized coordinates and $\dot{\varphi}_k = N_k \dot{q}_{i_k}$ is the angular velocity relative to the parent link.

Knee Joint Drive Kinematics

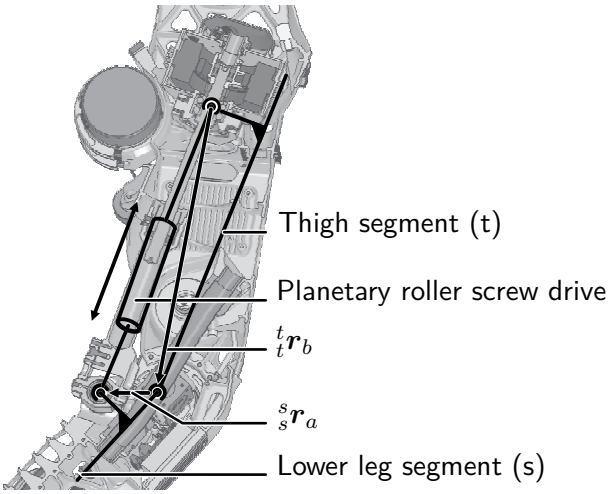


Figure 2.5: Knee joint drive kinematics

The knee joint is actuated by the nonlinear mechanism illustrated in Figure 2.5. The motor drives a planetary roller screw which converts the rotation into a linear motion. The linear motion is converted into the knee joint rotation by the 4-bar mechanism shown in Figure 2.5. The motor and attached roller screw rotate at high speed about the screw axis, while other components of angular velocity are much smaller. Therefore, the dynamic effects of the rotation orthogonal to the screw axis are neglected.

The screw's rotation is proportional to the nut's displacement l_{screw} and the screw lead P_{lead}:

$$ \varphi_{\text{screw}} = \frac{2\pi}{P_{\text{lead}}} l_{\text{screw}} \tag{2.14} $$

Calculating the relative kinematics therefore requires computing l_{screw}.

The vector from the pivot point on the thigh to the pivoting point on the lower leg ($\boldsymbol{r}_{\text{screw}}$) and the distance between these two points (l_{screw}) are given by:

$$\boldsymbol{r}_{\text{screw}} = {}^{t}_{t}\boldsymbol{r}_{b} + {}^{s}\boldsymbol{A}_{ts}\,{}^{s}_{s}\boldsymbol{r}_{a} \tag{2.15}$$

$$l_{\text{screw}} = \|\boldsymbol{r}_{\text{screw}}\| \tag{2.16}$$

Which yields:

$$\dot{l}_{\text{screw}} = \frac{\left({}^{s}\dot{\boldsymbol{A}}_{ts}\,{}^{s}_{s}\boldsymbol{r}_{a}\right)^{T}\boldsymbol{r}_{\text{screw}}}{l_{\text{screw}}} \tag{2.17}$$

$$\varphi_{\text{screw}} = \frac{2\pi l_{\text{screw}}}{P_{\text{lead}}} \tag{2.18}$$

$$\dot{\varphi}_{\text{screw}} = \frac{2\pi \dot{l}_{\text{screw}}}{P_{\text{lead}}} \tag{2.19}$$

$$\boldsymbol{\omega}_{rel} = \boldsymbol{e}_{z}\dot{\varphi}_{\text{screw}} \tag{2.20}$$

Finally, the relative rotational Jacobian is obtained by partial differentiation:

$$J_{Rrel,[i,j]} = \begin{cases} \frac{2\pi}{P_{\text{lead}} l_{\text{screw}}}\boldsymbol{r}^{T}_{\text{screw}}\left(\frac{\partial\,{}^{s}\boldsymbol{A}_{ts}}{\partial\, q_{\text{knee}}}\,{}^{s}_{s}\boldsymbol{r}_{a}\right) & \text{for} \quad i = 2 \wedge j = q_{\text{knee}} \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

Figure 2.6 shows the motor angle of the right knee joint drive as a function of the knee joint angle.

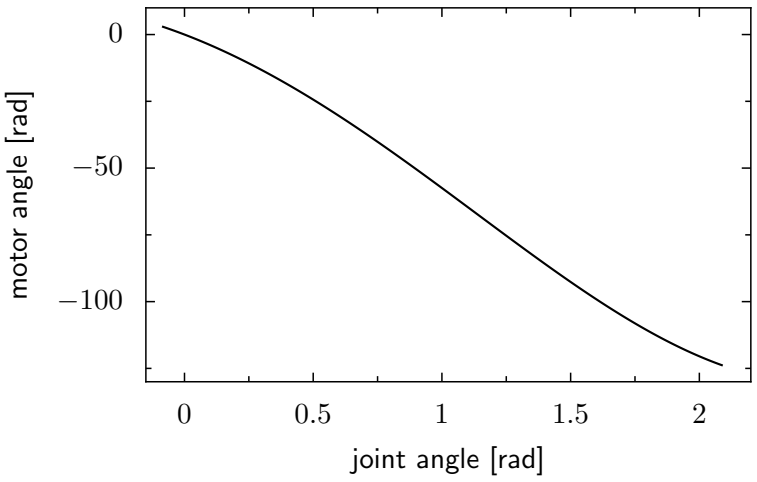


Figure 2.6: Nonlinear transmission of the right knee joint drive

Ankle Joint Drive Kinematics

Each ankle joint is driven by a pair of parallel, spatial slider-crank mechanisms, where the linear motion of a carriage is produced by planetary roller screws. The movement of the coupler link connecting the linear carriage to the foot segment relative to the lower leg is comparatively slow and the coupler link is very lightweight. Inertial effects due to the coupler link's motion are therefore neglected and only the

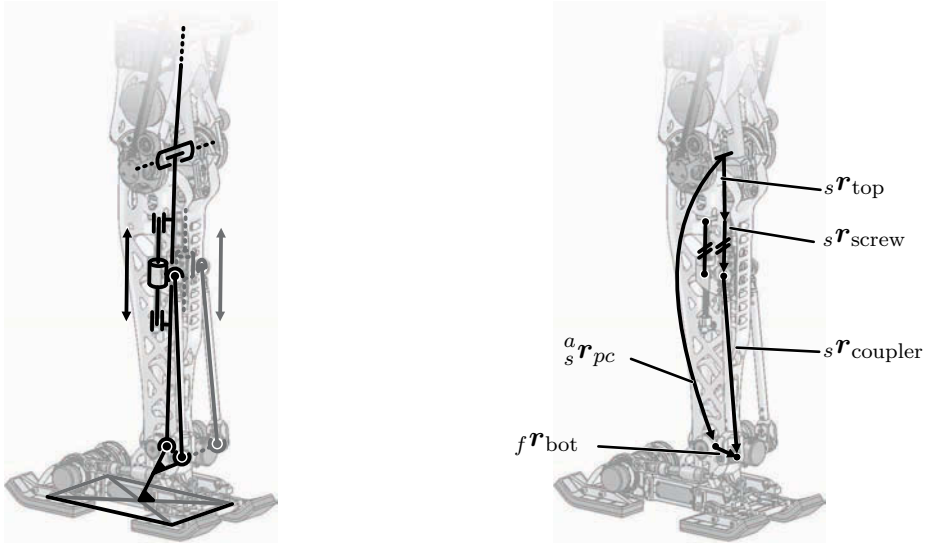


Figure 2.7: Kinematics of the ankle drive mechanism

much larger effects due to the screw's rotation are taken into account.

Similar to the procedure for the knee joint drive, solving the ankle joint kinematics requires expressing the displacement l_{screw} of the nut and rotation φ_{screw} of the screw as a function of the generalized coordinates $\boldsymbol{q}$.

Referring to Figure 2.7, the vector chain along the closed kinematic loop is given by:

$$ {}_s\boldsymbol{r}_{\text{coupler}} = \underbrace{{}^a_s\boldsymbol{r}_{pc} + \boldsymbol{A}_{sa}\left(\boldsymbol{A}_{af}\,{}_f\boldsymbol{r}_{\text{bot}} + {}^f_a\boldsymbol{r}_{pc}\right) - {}_s\boldsymbol{r}_{\text{top}}}_{{}_s\boldsymbol{r}_{\text{coupler},0}} - {}_s\boldsymbol{r}_{\text{screw}} \tag{2.22} $$

Here the letters s, a, f and c denote the shank, ankle, foot and coupler link respectively. ${}^f_a\boldsymbol{r}_{pc}$ is the vector from the ankle to the foot frame, which is not shown in Figure 2.7, since it is zero for Lola. However, this vector is non-zero for Johnnie.

Using the unit vector along the roller screw's major axis ${}_s\boldsymbol{e}_{\text{screw}}$, the known and constant length of the coupler link l_{coupler} and the relationship

$$ l^2_{\text{coupler}} = {}_s\boldsymbol{r}^T_{\text{coupler}}\,{}_s\boldsymbol{r}_{\text{coupler}}\,, \tag{2.23} $$

(2.22) and (2.23) can be solved for the displacement of the roller screw nut l_{screw}:

$$ l_{\text{screw}} = {}_s\boldsymbol{r}^T_{\text{coupler},0}\,\boldsymbol{e}_{\text{screw}} \overset{(+)}{} \sqrt{l^2_{\text{coupler}} - {}_s\boldsymbol{r}^2_{\text{coupler},0} + ({}_s\boldsymbol{r}^T_{\text{coupler},0}\,\boldsymbol{e}_{\text{screw}})^2} \tag{2.24} $$

where ${}_s\boldsymbol{e}_{\text{screw}} = {}_s\boldsymbol{r}_{\text{screw}}\,/l_{\text{screw}}$ is the unit vector along the screw's major axis. The minus sign gives the correct solution for Johnnie's and Lola's kinematics, which can be seen by studying the angle between the coupler link and the roller screw, or its cosine $\boldsymbol{e}^T_{\text{screw}}\boldsymbol{e}_{\text{coupler}}$, for both solutions.

From l_{screw} the rotation relative to the shank φ_{screw} and the relative angular velocity

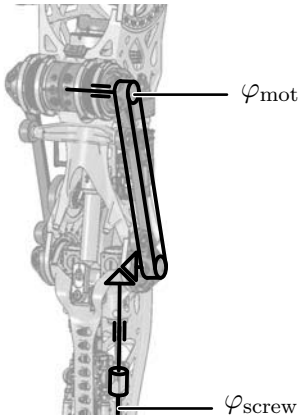


Figure 2.8: Upper part of Lola's ankle joint drive train

$\boldsymbol{\omega}_{\text{screw,rel}}$ are computed as:

$$\varphi_{\text{screw}} = \frac{2\pi l_{\text{coupler}}}{P_{\text{lead}}} \tag{2.25}$$

$$\dot{\varphi}_{\text{screw}} = \frac{2\pi \dot{l}_{\text{coupler}}}{P_{\text{lead}}} \tag{2.26}$$

$$\boldsymbol{\omega}_{\text{screw,rel}} = \boldsymbol{e}_z \dot{\varphi}_{\text{screw}} \tag{2.27}$$

The angular velocity, relative Jacobian, etc. are determined by straightforward differentiation:

$$\begin{aligned} \dot{l}_{\text{screw}} &= {}_s\dot{\boldsymbol{r}}^T_{\text{coupler},0}\, \boldsymbol{e}_s \\ &\quad - \frac{-\,{}_s\boldsymbol{r}^T_{\text{coupler},0}\, {}_s\dot{\boldsymbol{r}}_{\text{coupler},0} + ({}_s\boldsymbol{r}^T_{\text{coupler},0}\, \boldsymbol{e}_s)({}_s\dot{\boldsymbol{r}}^T_{\text{coupler},0}\, \boldsymbol{e}_s)}{\sqrt{l^2_{\text{coupler}} - {}_s\boldsymbol{r}^2_{\text{coupler},0} + ({}_s\boldsymbol{r}^T_{\text{coupler},0}\, \boldsymbol{e}_s)^2}} \end{aligned} \tag{2.28}$$

$$\dot{\varphi}_{\text{screw}} = \frac{2\pi \dot{l}_{\text{screw}}}{P_{\text{lead}}} \tag{2.29}$$

$$\begin{aligned} \frac{\partial\, \dot{l}_{\text{screw}}}{\partial\, \dot{\boldsymbol{q}}} &= \boldsymbol{e}_s^T \frac{\partial\ {}_s\dot{\boldsymbol{r}}_{\text{coupler},0}}{\partial\, \dot{\boldsymbol{q}}} \\ &\quad - \frac{-\,{}_s\boldsymbol{r}^T_{\text{coupler},0}\, \frac{\partial\, {}_s\dot{\boldsymbol{r}}_{\text{coupler},0}}{\partial\, \dot{\boldsymbol{q}}} + ({}_s\boldsymbol{r}^T_{\text{coupler},0}\, \boldsymbol{e}_s)(\boldsymbol{e}_s^T \frac{\partial\, {}_s\dot{\boldsymbol{r}}^T_{\text{coupler},0}}{\partial\, \dot{\boldsymbol{q}}})}{\sqrt{l^2_{\text{coupler}} - {}_s\boldsymbol{r}^2_{\text{coupler},0} + ({}_s\boldsymbol{r}^T_{\text{coupler},0}\, \boldsymbol{e}_s)^2}} \end{aligned} \tag{2.30}$$

For Lola, each roller screw is actuated by a motor mounted on the thigh, whose rotation is transferred to the screw via a synchronous belt drive and a bevel gear in the knee joint axis (cf. Figure 2.8). Therefore, the rotation of the driving motor depends not only on the ankle joint angles, but also on the knee joint angle:

$$\varphi_{\text{mot}} = \varphi_{\text{screw}} \pm \varphi_{\text{knee}} \tag{2.31}$$

$${}_{\text{mot}}\boldsymbol{\omega}_{\text{mot}} = \boldsymbol{e}_z\, (\dot{\varphi}_{\text{screw}} \pm \dot{\varphi}_{\text{knee}}) \tag{2.32}$$

The plus sign is valid for the motor driving the inner roller screw, the minus sign for the motor on the outer side.

With the knee joint angle as the j-th entry in $\boldsymbol{q}$, the relative rotational Jacobian for a motor driving an ankle joint is given by:

$$ {}_{\text{mot}}\boldsymbol{J}_{Rrel} = \frac{2\pi}{P_{\text{lead}}} \frac{\partial\, \dot{l}_{\text{screw}}}{\partial\, \dot{\boldsymbol{q}}} \pm \begin{pmatrix} \boldsymbol{0}_{3\times(j-1)} & \boldsymbol{e}_z & \boldsymbol{0}_{3\times(n-j)} \end{pmatrix} \tag{2.33} $$

Figure 2.9 shows the rotation of inner and outer roller screw of Lola's ankle joint mechanism as a function of the ankle joint adduction and flexion angles.

2.2.4 Equations of Motion for the Rigid Multibody System

Using the Newton-Euler-Jourdain principle [5, 126], the equations of motion (EoM) for the rigid MBS can be written as:

$$ \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{h} = \boldsymbol{Q}_{\text{ext}} \tag{2.34} $$

With the mass matrix

$$ \boldsymbol{M} = \sum_{i=0}^{N_{\text{body}}} \left\{ \begin{pmatrix} \boldsymbol{J}_{To} \\ \boldsymbol{J}_R \end{pmatrix}^T \begin{pmatrix} m\boldsymbol{E}_3 & m\tilde{\boldsymbol{r}}_{\text{CoG}}^T \\ m\tilde{\boldsymbol{r}}_{\text{CoG}} & \boldsymbol{I}_o \end{pmatrix} \begin{pmatrix} \boldsymbol{J}_{To} \\ \boldsymbol{J}_R \end{pmatrix} \right\}_i \tag{2.35} $$

and the vector of Coriolis, centrifugal and gravitational forces

$$ \begin{aligned} \boldsymbol{h} = \sum_{i=0}^{N_{\text{body}}} \Bigg\{ \begin{pmatrix} \boldsymbol{J}_{To} \\ \boldsymbol{J}_R \end{pmatrix}^T \Bigg[& \begin{pmatrix} m\boldsymbol{E}_3 & m\tilde{\boldsymbol{r}}_{\text{CoG}}^T \\ m\tilde{\boldsymbol{r}}_{\text{CoG}} & \boldsymbol{I}_o \end{pmatrix} \begin{pmatrix} \boldsymbol{J}_{Tqo} \\ \boldsymbol{J}_{Rq} \end{pmatrix} \dot{\boldsymbol{q}} \\ & + \begin{pmatrix} m\boldsymbol{E}_3 & m\tilde{\boldsymbol{r}}_s^T \\ m\tilde{\boldsymbol{r}}_s & \boldsymbol{I}_o \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{\omega}}\dot{\boldsymbol{r}}_o \\ \boldsymbol{0} \end{pmatrix} + \begin{pmatrix} m\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\boldsymbol{r}_{\text{CoG}} \\ \tilde{\boldsymbol{\omega}}\boldsymbol{I}_o\boldsymbol{\omega} \end{pmatrix} \\ & + \begin{pmatrix} m\boldsymbol{g} \\ m\tilde{\boldsymbol{r}}_{\text{CoG}}\boldsymbol{g} \end{pmatrix} \Bigg] \Bigg\}_i \end{aligned} \tag{2.36} $$

and the vector of impressed forces $\boldsymbol{Q}_{\text{ext}}$. N_{body} are the number of bodies in the MBS, $\boldsymbol{E}_3 \in \mathbb{R}^{3\times3}$ is a unit matrix, $\boldsymbol{I}_o$ the inertia matrix with respect to the FoR, $\boldsymbol{g}$ is the gravitational acceleration and $\tilde{\boldsymbol{a}}$ is a skew-symmetric matrix representing the cross product, i.e., $\tilde{\boldsymbol{a}}\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$. The impressed forces $\boldsymbol{Q}_{\text{ext}}$ include contact forces[6] acting on the feet, gear friction and motor torque.

2.3 Contact and Environment Models

The robot's only interaction with the environment is through its feet. Consequently, the global dynamics, i.e., linear and angular momentum for the entire system, are strongly influenced by contact dynamics, which makes accurate modeling essential for dynamics simulation and control.

6 For conciseness the term "force" is used to mean "force and/or torque" throughout this thesis.

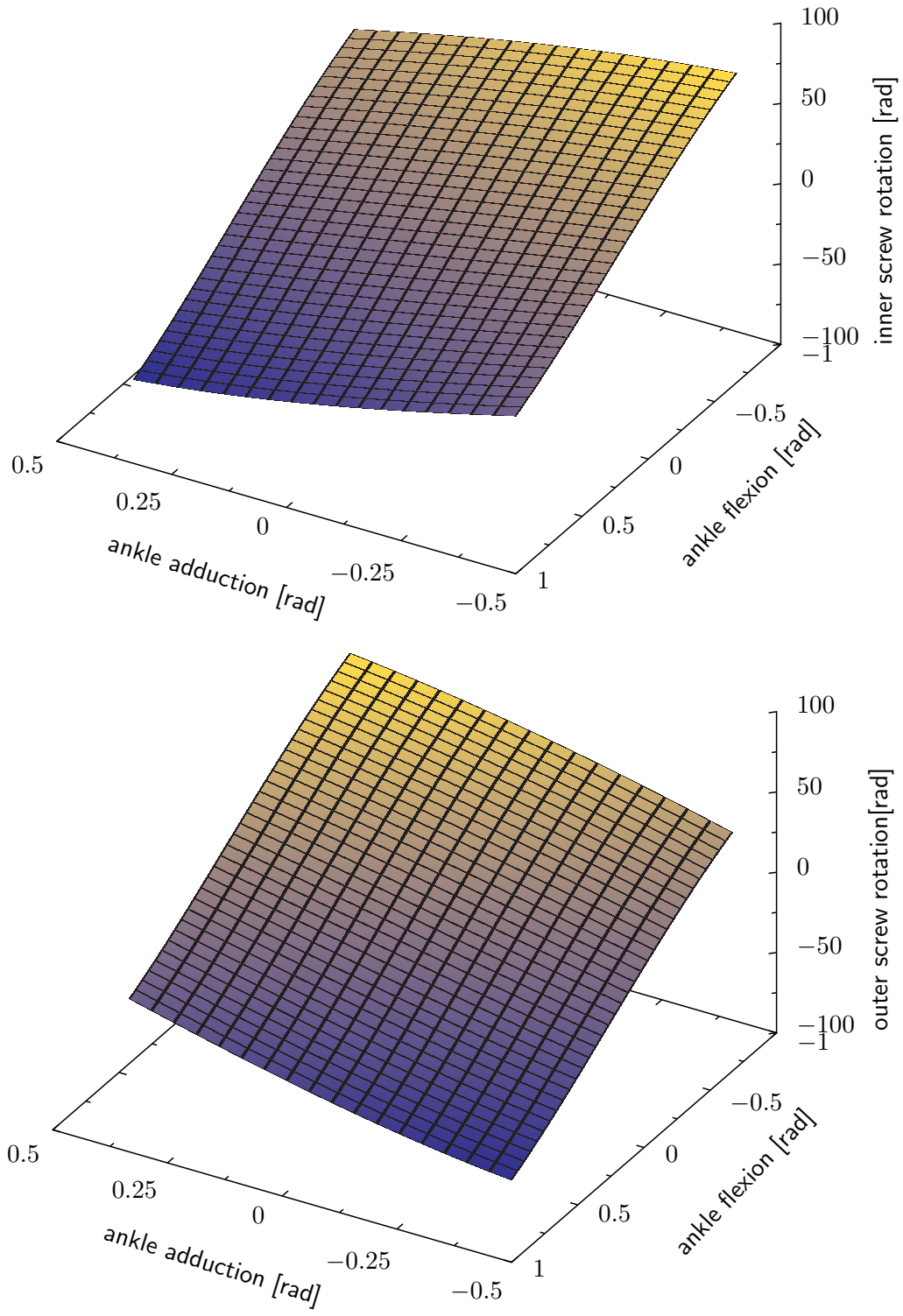


Figure 2.9: Nonlinear transmission of the right ankle joint drive

An in-depth presentation of contact mechanics of deformable bodies, including geometric relations, constraints and a range of discretization and solution approaches is given by Wriggers [133]. A detailed account of the dynamics of rigid multibody systems with unilateral contacts is given by Pfeiffer and Glocker [91].

2.3.1 Contact Dynamics

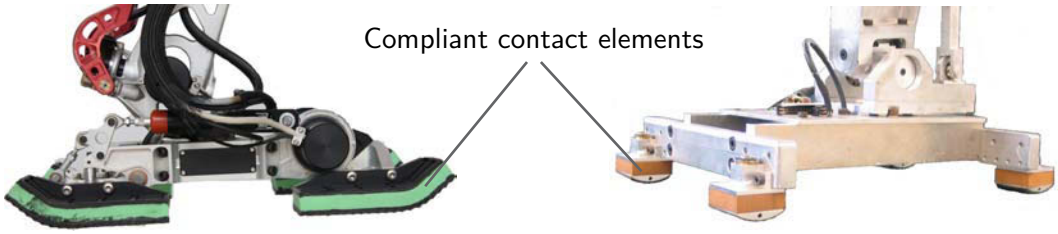


Figure 2.10: Feet of Lola (left) and Johnnie (right) with compliant contact elements

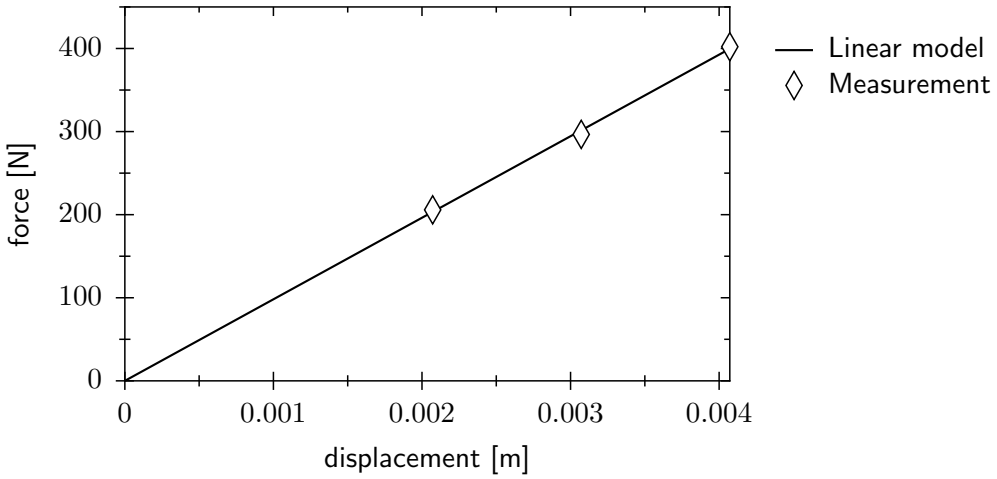


Figure 2.11: Static force-displacement relationship for one of Lola's contact elements. The solid line shows a least squares fit for a linear model.

Equations of Motion with Constraints

The feet of both Johnnie and Lola, shown in Figure 2.10, are equipped with viscoelastic contact elements. The dynamics of these elements are determined by the EoM for the compliant material and conditions for the unilateral contact with the environment.

Since the mass of the compliant material is very small, gravity and inertial forces are negligible when compared to external loads. The contact elements are therefore assumed to be massless. In quasi-static measurements, the contact elements show a remarkably linear force-displacement relationship for loads of up to a multiple of Lola's weight, as shown in Figure 2.11. Since experiments also show that the contact material exhibits significant damping, a linear viscoelastic material model is chosen. For numerical simulations, the contact elements are discretized either using the finite

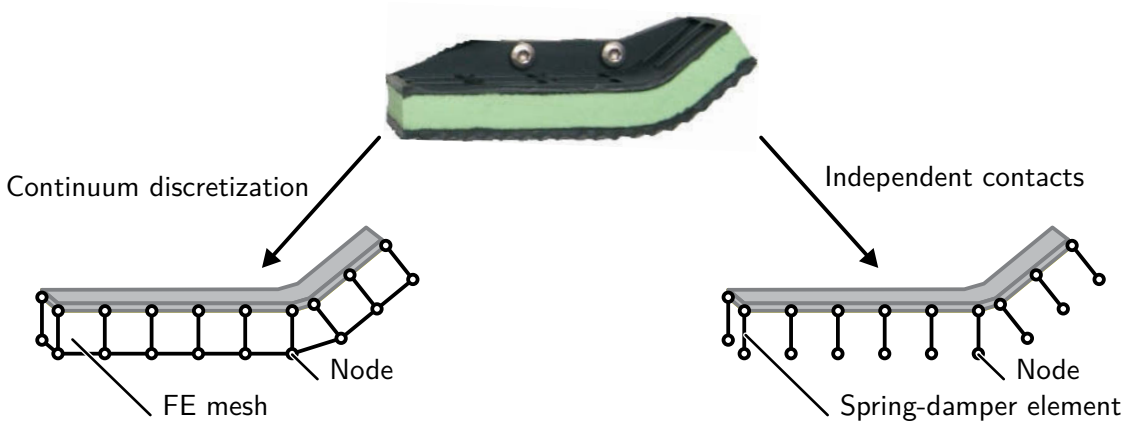


Figure 2.12: Discretization of contact layer using FEM and independent spring-damper systems

element method (FEM) or modeled as a set of independent spring-damper systems (cf. Figure 2.12). A geometrically linear FE model is used, but rigid body rotations and translations are taken into account by choosing the foot or toe frame as reference frame for node displacements. After discretization, the EoM for both model types can be written as

$$\boldsymbol{B}\dot{\boldsymbol{d}} + \boldsymbol{K}\boldsymbol{d} = \boldsymbol{f} \tag{2.37}$$

where $\boldsymbol{d}$ are node displacements, $\boldsymbol{f}$ node forces, $\boldsymbol{B}$ is a damping and $\boldsymbol{K}$ a stiffness matrix. $\boldsymbol{K}$ and $\boldsymbol{B}$ are diagonal for independent point contacts, while this is not the case for a general FE model. Meshes and stiffness matrices for the FE model are generated from CAD data using the commercial FEM program ANSYS[7]. Assuming Rayleigh-damping, $\boldsymbol{B}$ is then given by $\alpha\boldsymbol{K}$, with $\alpha > 0$ a scalar.

The contact element is rigidly attached to either a heel or toe segment. The force and torque acting from the contact element onto this segment can be calculated as

$$\boldsymbol{F}_{CE} = \sum_i \boldsymbol{f}_i \tag{2.38}$$

$$\boldsymbol{T}_{CE} = \sum_i \boldsymbol{\Delta r}_i \times \boldsymbol{f}_i \tag{2.39}$$

Here $\boldsymbol{f}_i$ is the force of the i-th free node and $\boldsymbol{\Delta r}_i$ the vector to this node from the segment's reference frame. The generalized force $\boldsymbol{Q}_{\text{cont}}$ acting on the MBS is then given by:

$$\boldsymbol{Q}_{\text{cont}} = \sum_i \boldsymbol{J}_{To,i}^T \boldsymbol{F}_{CE,i} + \boldsymbol{J}_{R,i}^T \boldsymbol{T}_{CE,i} \tag{2.40}$$

The unilateral contact between foot and environment imposes constraints on the motion of contact element nodes relative to the environment and on contact forces acting between foot and environment (cf. Figure 2.13). Normal forces λ_N must be positive and the nodes may not penetrate the environment, i.e., the minimal

7 http://www.ansys.com/products/default.asp

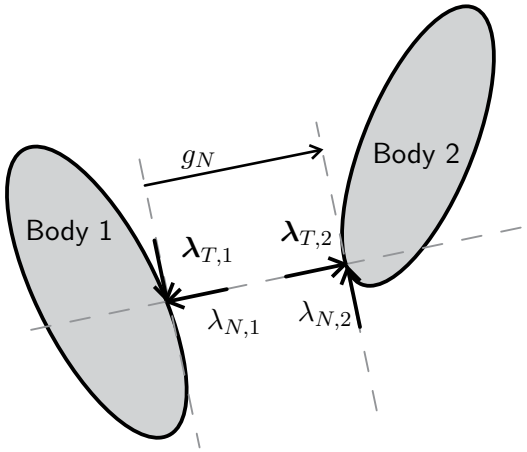


Figure 2.13: Unilateral contact between two bodies

distance g_N must be positive. Mathematically, this is expressed by the so-called Signiorini-Fichera condition:

$$\lambda_N \geq 0 \quad \wedge \quad g_N \geq 0 \quad \wedge \quad \lambda_N g_N = 0 \tag{2.41}$$

The tangential contact is modeled as Coulomb friction, which can be expressed by the following constraints for the relative tangential velocity $\dot{g}_T$ and tangential force $\boldsymbol{\lambda}_T$:

$$\lambda_N - \mu \left\| \boldsymbol{\lambda}_T \right\| \quad \geq 0 \quad \wedge \quad (\lambda_N - \mu \left\| \boldsymbol{\lambda}_T \right\|) \dot{\boldsymbol{g}}_T = 0 \tag{2.42}$$

Here the static and kinetic coefficients of friction are denoted by μ and assumed to be equal.

Due to the unilateral contact with friction, deformation rates $\dot{\boldsymbol{d}}$ and node forces $\boldsymbol{f}$ must be determined by solving (2.41), (2.42) and (2.37) simultaneously for the current robot state $\boldsymbol{q}, \dot{\boldsymbol{q}}$.

General Contact Solver

For a non-diagonal matrix $\boldsymbol{B}$, an approach based on exact regularization of the constraints was implemented for determining $\dot{\boldsymbol{d}}$ and $\boldsymbol{f}$. The method has its origins in non-smooth optimization and has been widely used for simulating non-smooth dynamical systems (see [22] and references therein). In the following the concept of the proximal point of a convex set C to a point $\boldsymbol{z}$ [102] is used. This is simply the closest point to $\boldsymbol{z}$ in $\mathscr{C}$:

$$\mathbf{prox}_C(\boldsymbol{z}) = \arg\min_{\boldsymbol{x}^* \in \mathscr{C}} \left\| \boldsymbol{z} - \boldsymbol{x}^* \right\|, \quad \boldsymbol{z} \in \mathbb{R}^n \tag{2.43}$$

Using this concept, the equality and inequality constraints are converted into a set of non-smooth equality constraints [21, 57]:

$$\lambda_{N,i} = \mathrm{prox}_{\mathscr{C}_N}(\lambda_{N,i} - r\dot{g}_{N,i}) \quad \forall i \in I_c \tag{2.44}$$

$$\boldsymbol{\lambda}_{T,i} = \mathbf{prox}_{\mathscr{C}_T(\lambda_N)}(\boldsymbol{\lambda}_{T,i} - r\dot{\boldsymbol{g}}_{T,i}) \quad \forall i \in I_c \tag{2.45}$$

Here $I_c = \{i | g_{N,i} = 0\}$ is the set of active contacts and $r > 0$ is a numerical constant. The sets $\mathscr{C}_T, \mathscr{C}_N$ respectively specify the admissible tangential and normal forces:

$$\mathscr{C}_N := \{x | x \geq 0\} \tag{2.46}$$

$$\mathscr{C}_T(N) := \{\boldsymbol{x} | \, \|\boldsymbol{x}\| \leq \mu N\} \tag{2.47}$$

The distance $g_{N,i}$ and relative velocity $\dot{g}_{N,i}, \dot{\boldsymbol{g}}_{T,i}$ of the i-th contact point to the environment can be calculated by a distance query using the environment model and the kinematics of the robot (cf. Section 2.3.2). Since $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and $\boldsymbol{d}$ are known at each time step and $\dot{\boldsymbol{d}}, \boldsymbol{f}$ are linked by the contact layer's EoM (2.37), equations (2.44) and (2.45) can be solved using the fixed-point iteration scheme shown in Algorithm 1.

Algorithm 1 Contact force and deformation for FE contact model

$I_c \leftarrow \emptyset$
for all nodes i **do**
  calculate gap function $g_{N,i}$
  if $\|g_{N,i}\| < \varepsilon_{\text{gap}}$ **then**
    add i to I_c
  end if
end for
repeat
  $\dot{\boldsymbol{d}} \leftarrow \boldsymbol{B}^{-1}(\boldsymbol{f} - \boldsymbol{K}\boldsymbol{d})$
  $\boldsymbol{f}_{\text{old}} \leftarrow \boldsymbol{f}$
  for all contacts i **do**
    if $i \in I_c$ **then**
      calculate relative velocity $\dot{g}_{N,i}, \dot{\boldsymbol{g}}_{T,i}$
      calculate contact force components $\lambda_{N,i}, \boldsymbol{\lambda}_{T,i}$
      calculate $r_{N,i}, r_{T,i}$
      $\lambda_{N,i} \leftarrow \text{prox}_{\mathscr{C}_N}(\lambda_{N,i} - r_{N,i}\dot{g}_{N,i})$
      $\boldsymbol{\lambda}_{T,i} \leftarrow \textbf{prox}_{\mathscr{C}_T(\lambda_N)}(\boldsymbol{\lambda}_{T,i} - r_{T,i}\dot{\boldsymbol{g}}_{T,i})$
    else
      $\boldsymbol{\lambda} \leftarrow (\boldsymbol{\lambda}_T^T, \lambda_N)^T = \boldsymbol{0}$
    end if
  end for
  update $\boldsymbol{f}$ using new $\boldsymbol{\lambda}_i$
until $\|\boldsymbol{f} - \boldsymbol{f}_{\text{old}}\| < \varepsilon_{\text{f}}$

The gap function is calculated using the environment model and the current value of the position states $\boldsymbol{q}, \boldsymbol{d}$. At each time step the iteration is started using the forces from the last time step. In order to reduce the numerical cost for calculating $\dot{\boldsymbol{d}}$, $\boldsymbol{B}^{-1}$ is calculated in a pre-processing step, so only a matrix-vector multiplication must be calculated at run-time.

The rate of convergence strongly depends on the r-factor. In [22] different strategies are presented for selecting r in systems with impacts. Starting from the observation that the displacement of the i-th node has the strongest influence on the node force $\boldsymbol{f}_i$, the following local strategy for selecting $r_{N,i}, r_{T,i}$ was developed:

$$r_{N,i} = \gamma_N / (\boldsymbol{e}_{N,i}^T \boldsymbol{B}_{ii} \boldsymbol{e}_{N,i}) \tag{2.48}$$

$$r_{T,i} = \gamma_T / \sqrt{(\boldsymbol{e}_{T1,i}^T \boldsymbol{B}_{ii} \boldsymbol{e}_{T1,i})^2 + (\boldsymbol{e}_{T2,i}^T \boldsymbol{B}_{ii} \boldsymbol{e}_{T2,i})^2} \tag{2.49}$$

Here $\boldsymbol{B}_{ii}$ is the 3×3 sub-matrix of $\boldsymbol{B}$ determining the coupling between $\boldsymbol{f}_i$ and $\dot{\boldsymbol{d}}_i$ and $\boldsymbol{e}_{N,i}$, $\boldsymbol{e}_{T1,i}$, $\boldsymbol{e}_{T2,i}$ are unit vectors in normal and tangential directions of the contact plane, respectively. The scalars $\gamma_{T/N} \in (0, 2)$ depend on the structure of $\boldsymbol{B}$ and are chosen by trial and error to assure fast convergence and avoid divergence of the iteration. In numerical experiments values of $\gamma_{T/N} \approx 0.5 \ldots 0.9$ worked well.

Decoupled Contact Solver

For the special case of independent contacts, a simpler and more efficient solution is possible, since there are only a small number of possible contact states for each node. In this situation, explicitly checking different contact states is more efficient than the fixed-point scheme presented above.

2.3.2 Environment Model and Distance Computation

Simulating stepping over obstacles, climbing stairs or the effects of an uneven floor requires modeling complex environment geometries. For the compliant models described in the previous section, such an environment can be approximated with arbitrary precision using a triangular surface mesh. This approach is especially well suited for objects typically used in experiments such as boards, stairs or boxes.

While complex environment geometries can be simulated, dynamic environments that are modified by the robot's actions are not considered. That is, all objects in the environment are assumed to be rigid and fixed with respect to the inertial reference frame while in contact with the robot. However, objects may move, appear or disappear when they are not in contact with the robot.

Environment Model

More formally, the environment $\mathcal{E}$ is modeled as a set of rigid objects $\mathcal{O}_i$, whose surfaces are approximated by triangular meshes (cf. Figure 2.14). Each object can be described by a set of triangles $\mathcal{T}_{ij}$, which in turn are defined by their vertices $\boldsymbol{v}_{ijk}$:

$$\mathcal{E} := \{\mathcal{O}_i\}, \qquad i \in \{0 \ldots n_{\mathcal{O}} - 1\} \tag{2.50}$$
$$\mathcal{O}_i := \{\mathcal{T}_{ij}\}, \qquad j \in \{0 \ldots n_{\mathcal{T},i} - 1\} \tag{2.51}$$
$$\mathcal{T}_{ij} := \{\boldsymbol{v}_{ijk}\}, \qquad k \in \{0 \ldots 2\} \tag{2.52}$$

This model is sufficiently general to cover indoor and outdoor environments, including obstacles with complex geometries. At the same time, high-quality meshes can easily be generated using many three-dimensional (3-D) modeling or computer aided design (CAD) packages. For the work presented here, CAD models were generated using CATIA V5[8] and exported in a standard 3-D geometry file format.

8 A 3-D CAD system by Dassault Systèmes: http://www.3ds.com/products/catia.

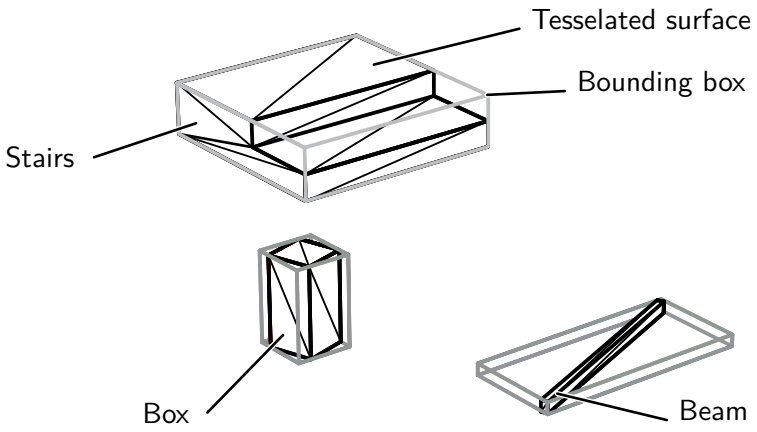


Figure 2.14: Example of an environment model used in simulations

The simulation system currently supports STL[9] and VRML[10] files.

Distance Computation

Calculating contact forces requires knowledge of potential contact points and surface normals. Contact determination, also known as collision or interference detection, is central not only to dynamics simulation, but also to other fields such as robot motion planning [83], computer graphics, CAD and others. In many scenarios contact determination is a major computational bottleneck. Therefore, much research has gone into developing efficient algorithms for this task. Surveys of the field are given in [40, 59]. In motion planning and computer graphics polygonal models are dominant, since they enable the approximation of arbitrarily complex geometries and hardware-accelerated rendering is enabled by modern graphics processing units (GPUs). For the simulation of rigid multibody systems with (unilateral) contacts, the focus often is on analytical methods for special cases, such as plane-sphere or point-line contacts [91]. This is due to the fact that in machine dynamics a good approximation of the contact position is often known in advance and the (local) surface geometry can be described accurately using geometric primitives such as cylinders or planes. Finally, for rigid body contacts the contour geometry and curvature must be accurately modeled in order to avoid artificial impulse effects during rolling. This is not the case for the contact model described above, since it is compliant *and* massless.

For the model presented here, all contact forces are assumed to act at the nodes, i.e., collisions between the environment and edges or surfaces of the FE-mesh are not considered. Then, contact determination amounts to checking triangular meshes for collision with contact element nodes. This is similar to the standard problem in computational geometry in which a number of objects approximated by polygon meshes are checked for collision. In both cases the basic operation (calculating the distance from a triangle to a point or a second triangle) is very simple. However, due to the large number of point/triangle or triangle/triangle pairs, the computational

9 STL is a format originating in stereo lithography. A definition can be found at `http://www.ennex.com/~fabbers/StL.asp`.

10 Virtual Reality Modeling Language is a file format for 3-D models standardized as ISO/IEC 14772-1:1997. The VRML loading code is based on work by Markus Schwienbacher.

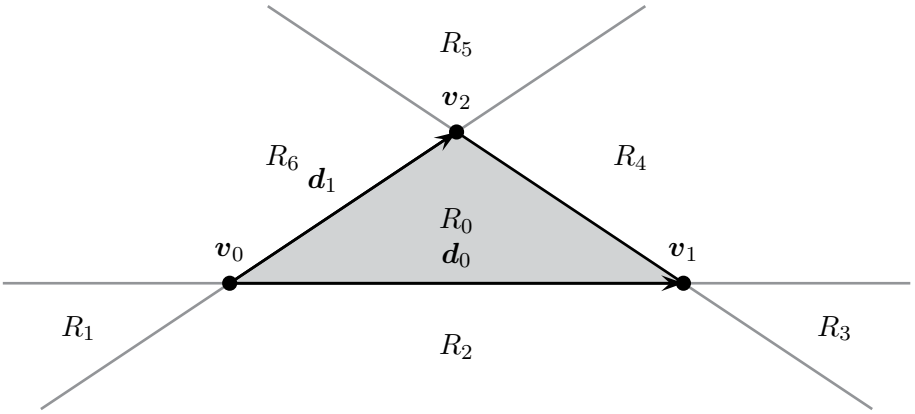


Figure 2.15: Enumeration of possible positions of $\boldsymbol{p}_\perp$ in or around a triangle

burden can be significant. A number of methods have been developed that aim at reducing the number of necessary checks [40, 59, 83]. The basic idea is to introduce a hierarchy of increasingly simpler models bounding the original, complex geometry from the outside. Then, the more complex models must only be checked for collision if a collision with the simpler bounding volume was detected. In this thesis one axially aligned bounding box per object $\mathcal{O}$, denoted by $\mathcal{B}(\mathcal{O})$, is used.

The shortest distance from $\boldsymbol{p}$ to an object $\mathcal{O}$ is denoted by $d(\boldsymbol{p}, \mathcal{O})$ and the closest point to $\boldsymbol{p}$ on $\mathcal{O}$ as $\boldsymbol{p}_{\min}(\boldsymbol{p}, \mathcal{O})$. The shortest distance from $\boldsymbol{p}$ to the environment and the normal $\boldsymbol{n}_\mathcal{E}$ to the contact plane can then be calculated using Algorithm 2.

Algorithm 2 Minimal distance computation between a point $\boldsymbol{p}$ and $\mathcal{E}$

```
d_min ← ∞
p_min ← None
n_min ← None
for all O_i ∈ E do
    if p ∈ B(O_i) then
        for all T_ij ∈ O_i do
            calculate d(T_ij, p) and p_min(p, T_ij)
            if d(T_ij, p) < d_min then
                d_min ← d(T_ij, p)
                p_min ← p_min(p, T_ij)
                n_min ← n(T_ij)
            end if
        end for
    end if
end for
d(p, E) ← d_min
p_min(p, E) ← p_min
n_E ← n_min
```

Depending on the relative position of $\boldsymbol{p}$ with respect to $\mathcal{T}_{ij}$, $\boldsymbol{p}_{\min}$ may be inside the triangle, on an edge or on a vertex. The algorithm for calculating $\boldsymbol{p}_{\min}$ consists of first determining which region $R_0 \dots R_6$ the projection $\boldsymbol{p}_\perp$ of $\boldsymbol{p}$ onto the plane defined by $\mathcal{T}_{ij}$ lies in. Figure 2.15 illustrates the different regions $\boldsymbol{p}_\perp$ can lie in.

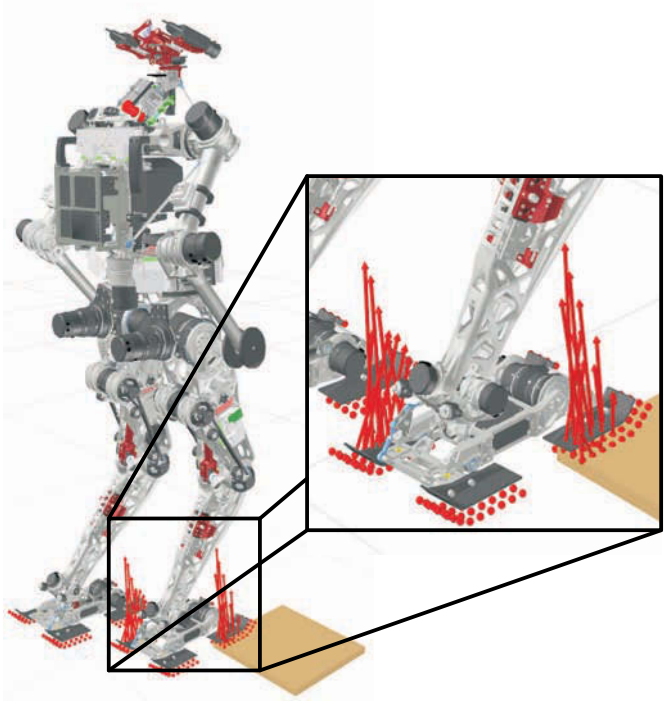


Figure 2.16: Simulation of Lola walking over a board using FE contact and polygonal environment model. Nodes are shown as red spheres, contact forces as red arrows.

The distance is computed by a simple point-to-plane, point-to-line or point-to-point distance computation.

In the following, indices ij for triangles are omitted for the sake of clarity. $\boldsymbol{p}_\perp$ is defined as the point in the plane $\mathcal{P}(\mathcal{T})$ with the minimal distance to $\boldsymbol{p}$:

$$\boldsymbol{p}_\perp = \underset{\boldsymbol{r}\in\mathcal{P}}{\operatorname{argmin}} \quad \frac{1}{2}(\boldsymbol{r}-\boldsymbol{p})^2 \tag{2.53}$$

Since $\boldsymbol{r} \in \mathcal{P}$, it can be expressed as a linear combination of two edge vectors of $\mathcal{T}$:

$$\boldsymbol{r} = \boldsymbol{v}_0 + (\boldsymbol{v}_1 - \boldsymbol{v}_0)s_0 + (\boldsymbol{v}_2 - \boldsymbol{v}_0)s_1 = \boldsymbol{v}_0 + \boldsymbol{D}\boldsymbol{s} \tag{2.54}$$

This leads to the optimization problem

$$\frac{1}{2}(\boldsymbol{v}_0 + \boldsymbol{D}\boldsymbol{s} - \boldsymbol{p})^2 \rightarrow \text{min!} \tag{2.55}$$

which can be solved for $\boldsymbol{s}$:

$$\boldsymbol{s} = \left(\boldsymbol{D}^T\boldsymbol{D}\right)^{-1}\boldsymbol{D}^T(\boldsymbol{p}-\boldsymbol{v}_0) = \boldsymbol{S}(\boldsymbol{p}-\boldsymbol{v}_0) \tag{2.56}$$

Since $\boldsymbol{S}$ can be precomputed, calculating $\boldsymbol{s}$ during time integration is quite efficient. Which region $\boldsymbol{p}_\perp$ lies in can be determined by studying $\boldsymbol{s}$. With reference to Figure 2.15, the different cases are summarized in Table 2.2. After the contact configuration has been determined, the actual distance computation can be calculated using elementary vector geometry.

An example of a simulation using the proposed environment and FE-contact models, in which Lola is walking over a board, is shown in Figure 2.16.

Table 2.2: Cases in the point-triangle distance computation

Region	Location of $\boldsymbol{p}_{\min}(\mathcal{E}, \boldsymbol{p})$	Condition
R_0	Inside triangle	$s_0 \geq 0 \wedge s_1 \geq 0 \wedge s_0 + s_1 \leq 1$
R_1	On vertex	$s_0 < 0 \wedge s_1 < 0$
R_2	On edge	$s_0 > 0 \wedge s_0 \leq 1 \wedge s_1 < 0$
R_3	On vertex	$s_0 > 1 \wedge s_1 < 0$
R_4	On edge	$s_0 \geq 0 \wedge s_1 \geq 0 \wedge s_0 + s_1 > 1$
R_5	On vertex	$s_0 < 0 \wedge s_1 > 1$
R_6	On edge	$s_0 < 0 \wedge s_1 > 0 \wedge s_1 \leq 1$

2.4 Drives

The robots Johnnie and Lola are actuated by drives based on electric motors and harmonic drive gears or planetary roller screws. Dynamics of the drives (besides rigid body mechanics) are integrated into the MBS as generalized forces $\boldsymbol{Q}_{\text{drive}}$. The following effects are taken into account:

1. Electrical motor dynamics ($\boldsymbol{Q}_{\text{mot}}$)
2. Gear friction ($\boldsymbol{Q}_{\text{gear,frict}}$)
3. Gear elasticity ($\boldsymbol{Q}_{\text{gear,elast}}$)

2.4.1 Electrical Motor Dynamics

Johnnie is driven by brushed DC motors, while PMSM are used in Lola. Strictly speaking, the simulation model for Lola should therefore include all three winding currents. However, assuming the field-oriented motor control is operating properly, the magnetizing currents $\boldsymbol{I}_d$ are controlled to zero [108] and the torque producing currents $\boldsymbol{I}_q = \boldsymbol{I}$ are used for speed or position control. In fact, this is a common simplification in simulating robotic systems also proposed by Siciliano [111]. Measurements of I_q and I_d for Lola's knee motor while walking on the spot, shown in Figure 2.17, confirm that this is a reasonable assumption. Both brushed DC motors and PMSM can then be described by the same differential equations:

$$\boldsymbol{L}\dot{\boldsymbol{I}} + \boldsymbol{R}\boldsymbol{I} + \boldsymbol{k}_M\boldsymbol{\omega}_{\text{rot}} = \boldsymbol{U} \tag{2.57}$$

Here $\boldsymbol{U}$ is the armature voltage, $\boldsymbol{\omega}_{\text{rot}}$ the rotor's angular velocity and $\boldsymbol{L}$, $\boldsymbol{R}$ and $\boldsymbol{k}_M$ are diagonal matrices of inductance, resistance and motor back-EMF[11] constants,

11 Electro Motive Force

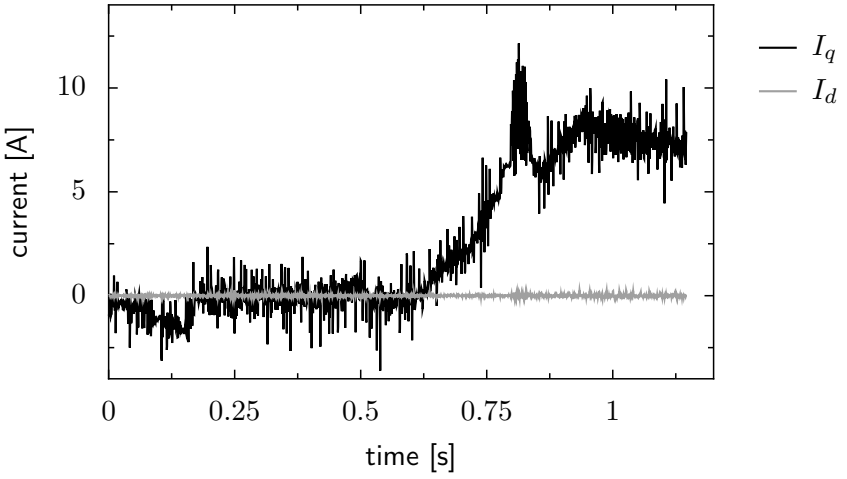


Figure 2.17: Active and reactive currents of the knee motor measured in an experiment with Lola walking on the spot

respectively. The generalized actuator forces are then given by:

$$\boldsymbol{Q}_{\text{mot}} = \sum_i \boldsymbol{J}_{R,i}^T \boldsymbol{e}_z k_{M,i} I_i \tag{2.58}$$

2.4.2 Gear Friction

Friction in harmonic drive gears and roller screws is modeled by nonlinear force laws that output a friction torque T_f acting on the motor shaft as a function of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. Friction is integrated into the MBS model by projecting T_f into the space of generalized coordinates via the motor shaft rotational Jacobians. Summing over all gears gives the vector of generalized gear friction forces $\boldsymbol{Q}_{\text{gear,frict}}$ to be added to the EoM:

$$\boldsymbol{Q}_{\text{gear,frict}} = \sum_i \boldsymbol{J}_{R,i}^T \boldsymbol{e}_z T_{f,i} \tag{2.59}$$

Harmonic Drive Gears

Modeling friction in harmonic drive gears is quite complex, since it depends not only on load, speed and temperature, but also on manufacturing and mounting errors and flange stiffness. Accordingly, most work in this area is based on extensive measurements of friction torques for different operating conditions [53, 87, 114]. Taghirad identifies compliance, load and speed dependent friction as the dominant effects in harmonic drive gears [115]. In accordance with claims by Harmonic Drive, he states that there is no stiction, but rising friction at low velocities.

In [53, 114] gear friction is modeled as load dependent Stribeck friction. Peer uses a linear model in combination with an experimentally determined friction loss table [87].

While models based on experimental data are more accurate, they obviously cannot be used for simulations during the development process of a new robot. Furthermore, since Lola and Johnnie do not have joint torque sensors, directly identifying friction

models is impossible. Therefore, an approach based on fitting friction models to catalog data was chosen. For all models, a constant ambient temperature of 20 °C is assumed.

Rossmann and Löffler model harmonic drive friction using three terms [60, 103]:

1. No-load starting torque/no-load backdriving torque $T_{f,0}$
2. Load dependent friction $T_{f,s}$
3. Viscous friction $T_{f,v}$

In this model, the friction torque T_f is calculated using the following equation:

$$T_f = T_{f,0} + T_{f,s} + T_{f,v} = -\operatorname{sign}(\omega)(T_{f,0} + \mu \|T_l\|) - b\omega - g\omega^3 \tag{2.60}$$

The load torque T_l is defined as the torque acting at the output shaft divided by the gear ratio N. Model parameters $T_{f,0}, \mu, b, g$ are determined from catalog data by least squares fitting.

Rossmann achieves a relatively good fit to catalog data for a small gear box (HDUC-14) using this approach. However, it proved impossible to get good fits for many gears used in Lola using the friction model (2.60). Figure 2.18 shows the efficiency for a HFUC-25-100 harmonic drive[12] as a function of load torque and wave generator speed.

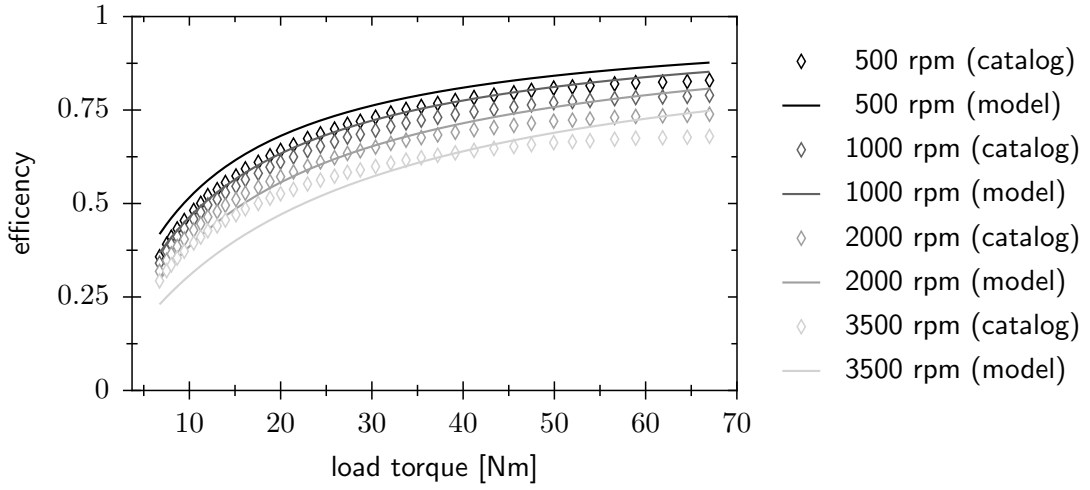


Figure 2.18: Comparison of model prediction to catalog data for a HFUC-25-100 gear using Rossmann's friction model (2.60)

The modeling error is quite large, because a *coupled load and speed dependent* term is missing in (2.60), i.e., $\partial^2 T_f / \partial T_l \partial \omega \equiv 0$. The error is larger for larger load torques, which is the typical operating range for the robot.

To capture the coupled load and speed dependency, an additional term $T_{f,sv} = -\gamma\omega \|T_l\|$ is added. Furthermore, the cubic term in (2.60) is omitted, since least

12 Nomenclature follows the Harmonic Drive catalog: Series-Size-Ratio.

squares fits consistently yielded very small values for g well below the accuracy of the friction model (often, $g \leq 10^{-10}$). Also, most of the time $g \leq 0$, leading to $T_{f,v}\omega \geq 0$ for angular velocities above the largest velocity used for least squares fitting. This is physically impossible and can lead to efficiencies larger than one, i.e., the model starts to generate energy instead of dissipating it.

The author therefore proposes the following model for harmonic drive friction:

$$\begin{aligned} T_f &= T_{f,0} + T_{f,s} + T_{f,v} + T_{f,sv} \\ &= -\operatorname{sign}(\omega)(T_{f,0} + \mu \|T_l\|) - b\omega - \gamma \|T_l\| \omega \\ &= -\operatorname{sign}(\omega)(T_{f,0} + \mu \|T_l\|) - (b + \gamma \|T_l\|)\omega \end{aligned} \tag{2.61}$$

The resulting efficiency is compared to catalog data in Figure 2.19. Using this model, the maximum approximation error is decreased from $7.91 \cdot 10^{-2}$ to $2.33 \cdot 10^{-2}$, or to below 30% of the original error.

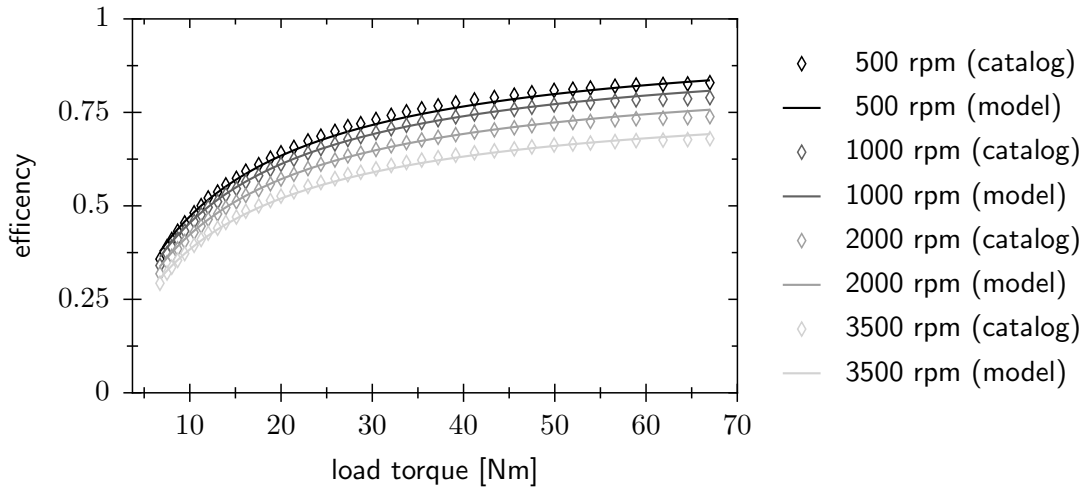


Figure 2.19: Comparison of model prediction to catalog data for a HFUC-25-100 gear using the newly developed friction model

Friction data is only available for T_l up to the nominal load and wave generator speeds up to 3500 rpm, while the robot is operated at both higher loads and speeds. When friction parameters are calculated by simple least squares fitting, it is possible for the model to predict efficiencies ≥ 1 at higher loads and/or speeds. A physically consistent friction model is guaranteed by adding appropriate inequality constraints to the parameter fitting procedure:

$$\begin{aligned} &\sum_i \left(\eta_{i,\text{catalog}} - \eta_{i,\text{model}}\right)^2 \rightarrow \min! \\ &\mu \geq 0 \\ &b \geq 0 \\ &T_0 \geq 0 \end{aligned} \tag{2.62}$$

Here $\eta_{i,\text{catalog}}$ and $\eta_{i,\text{model}}$ are efficiencies calculated from catalog data and the friction model. The parameter optimization problem (2.62) was solved using Matlab's

`fmincon` function. Friction parameters and a comparison to catalog data for all gears used in Lola is given in Appendix C.

Evaluating (2.61) requires knowledge of both the relative motor shaft velocity ω and the load torque T_l. The angular velocity ω can easily be calculated from generalized coordinates and velocities. The load torque T_l is determined from the torque balance at the gear box:

$$T_l = T_f + T_{\text{mot}} \tag{2.63}$$

Using (2.61) this yields:

$$T_l = -\operatorname{sign}(\omega) T_{f,0} - b\omega - (\operatorname{sign}(\omega)\mu + \omega\gamma) \|T_l\| + T_{\text{mot}} \tag{2.64}$$

This is solved for T_l by studying the cases for $T_l > 0$, $T_l < 0$ and $T_l = 0$.

Roller Screw Drives

Usually, friction in roller screw or ballscrew drives is modeled by some combination of static, viscous and Stribeck friction. Papadopoulos models friction in roller screws using a classical friction model consisting of static, viscous and Stribeck friction [84]. Riebe uses a similar approach to modeling friction in ballscrew drives of a hexapod [100]. In both cases, parameters for the friction models are determined experimentally. According to [50], roller screw-based actuators also exhibit significant load dependent friction. Since the Stribeck-effect is only relevant for low-speed operation, the friction in the robot's roller screws could be described by the model used for harmonic drive gears. Unfortunately, the only data provided by the manufacturer of the roller screws used for Lola are the direct (η_D) and indirect (η_I) efficiencies listed in Table 2.3. Using the Heaviside step function $H(\cdot)$, this data can be represented by the following piecewise constant function:

$$\eta = \eta_D + (\eta_I - \eta_D) H(\omega T_{\text{mot}}) \tag{2.65}$$

Using friction model (2.61), the screw efficiency for $T_l > 0 \wedge \omega > 0$ is calculated as:

$$\eta_D = \frac{T_l}{T_m} = \frac{-b\omega - T_0 + T_{\text{mot}}}{T_{\text{mot}}(1 + \mu + \gamma\omega)} \tag{2.66}$$

Since η_D is constant, all parameters except μ must vanish, leaving

$$\eta_D = \frac{1}{1+\mu} \tag{2.67}$$

and therefore $T_f \propto T_l$. That is, there is only load dependent friction. Consequently, the friction torque is given by:

$$T_f = -\operatorname{sign}(\omega) \|(1-\eta) T_{\text{mot}}\| \tag{2.68}$$

For the ankle joint drives, the efficiencies η_D, η_I are modified to take the timing

Table 2.3: Friction model parameters for roller screw-based drives

Part	η_D	η_I
Knee joint roller screw	0.860	0.858
Ankle joint roller screw	0.855	0.839
Ankle joint timing belt	0.98	0.98

belt efficiency into account.

Experiments with Lola revealed significant differences in the friction characteristics of roller screws. Therefore, a simple model seems to be the only sensible approach to simulation and modeling, especially during the design of a new robot.

2.4.3 Gear Elasticity

Since planetary roller screws are very stiff, their compliance is neglected. Compliance in harmonic drive gears is significantly higher, even if it is small in absolute terms. In combination with link-side position sensing, compliance can lead to high frequency vibrations or even instability of the joint servo loop. If, on the other hand, encoders mounted on the motor shaft are used, gear elasticity degrades the accuracy of joint position control.

In order to study such phenomena, gear elasticity is modeled using a progressive and piecewise linear torsion/torque relationship [30]:

$$T_{\text{gear,elast}} = \begin{cases} K_0 \Delta\varphi & \text{for} \quad T_{\text{gear,elast}} < T_1 \\ T_1 + K_1 \Delta\varphi & \text{for} \quad T_1 \leq T_{\text{gear,elast}} < T_2 \\ T_2 + K_2 \Delta\varphi & \text{for} \quad T_{\text{gear,elast}} \geq T_2 \end{cases} \tag{2.69}$$

Figure 2.20 shows this model for some harmonic drive gears used in Lola.

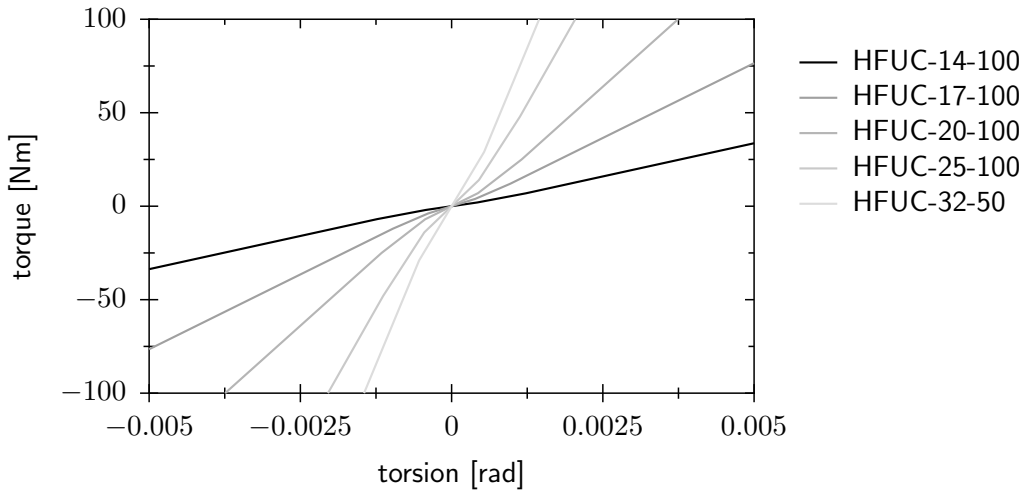


Figure 2.20: Harmonic drive gear component set elasticity according to [30]

As with gear friction, a vector of generalized forces is added to the MBS:

$$\boldsymbol{Q}_{\text{gear,elast}} = \sum_{\text{motors}} \boldsymbol{J}_R^T \boldsymbol{e}_z T_{\text{gear,elast}} + \sum_{\text{segments}} -\boldsymbol{J}_R^T \boldsymbol{e}_z T_{\text{gear,elast}} \tag{2.70}$$

2.5 Sensor Models

In order to simulate the complete system including the walking controller, sensor data must be calculated from the system state given by the simulation model. To obtain realistic simulation results, the sensor models should include sensor dynamics, signal quantization, dead-time and sensor noise. Also, there should be the possibility of introducing typical measurement errors into the simulation, in order to study the controller's robustness against such errors and to determine the necessary sensing accuracy and resolution during the design phase.

2.5.1 Joint Sensors

Incremental encoders mounted on the motor shafts are used in Johnnie, while Lola has additional absolute single turn encoders mounted on the output shafts. Both robots have high resolution encoders — incremental encoders in Lola's main leg joints have 11520 counts, Johnnie's have 2000 and Lola's absolute encoders have a resolution of either 16 or 17 bit. Therefore, the quantization error is well below the joint servo tracking error during walking. However, the quantization error is significantly higher for angular velocities calculated by finite differences. For a measurement with a resolution of Q bits, the maximum quantization error at the position level is $\Delta\varphi = \frac{2\pi}{2^Q - 1}$, or approximately 10^{-4} rad for a 16 bit sensor. Calculating the velocity by finite differences for a sampling rate T, the maximum error is given by $\Delta\dot{\varphi} = \frac{2\pi}{(2^Q - 1)T}$, which is approximately 0.1 rad/s for a 16 bit sensor at 1 kHz. Quantization errors are therefore included in the simulation model.

While the position sensors themselves are very accurate, a static bias is introduced due to kinematic calibration errors. Experiments have shown that such errors can strongly degrade walking performance. A static bias can therefore be set in the simulation model, in order to study the robustness of the walking control to such measurement errors. The overall model of joint angle sensors is shown in Figure 2.21.

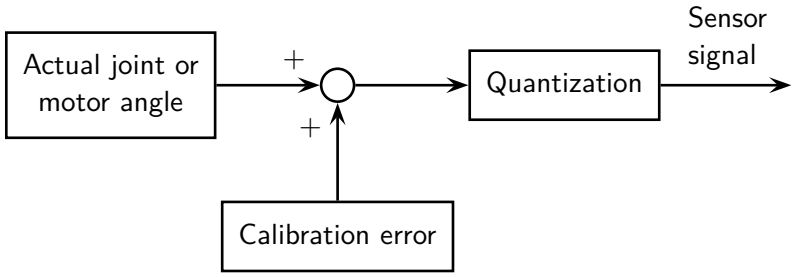


Figure 2.21: Simulation model for joint angle sensors

2.5.2 Force/Torque Sensors

The robot's six-axis force/torque sensors (FTS) are mounted between foot and ankle joint. Figure 2.22 shows the FTS used for Lola. Forces are measured by strain gauges integrated into the FTS, i.e., the sensors measure forces transmitted through the FTS and not the actual contact forces. The measured forces include contact forces as well as gravitational and inertial forces from the foot segment below the FTS. As part of the initialization procedure for the robot, the FTS signal is set to zero while the robot is hanging in the air and its feet are parallel to the ground. This adds a constant offset to the signal, so the true contact forces are obtained when the robot is standing on the ground (cf. "FTS bias compensation" in Figure 2.23). During walking, however, when the feet are accelerated or not parallel to the ground, this procedure no longer reproduces the true contact forces and torques since the constant offset added for bias compensation is no longer equal to the measurement error.

Exactly reproducing the sensor signal would require an elastic model of the sensor body, which would unacceptably increase simulation times. As an approximation, the constraint forces between foot and ankle joints are used for calculating the force sensor signal in the dynamics simulation and the same procedure for bias compensation is used.

Additionally, anti aliasing filters, quantization and noise are included. A schematic representation of the FTS model is shown in Figure 2.23.

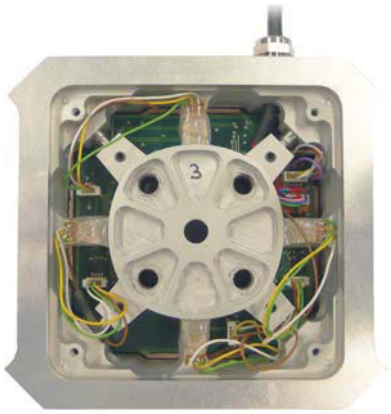


Figure 2.22: Lola's six axis force/torque sensor (bottom view)

2.5.3 Inertial Measurement Unit

An inertial measurement unit (IMU) is mounted on the upper body of both robots, allowing estimation of angular velocity, orientation and linear acceleration based on gyroscopes and accelerometers. The sensors are arranged in a so-called strap-down configuration, i.e., they are fixed with respect to the robot. An overview of inertial measurement and navigation can be found in [113].

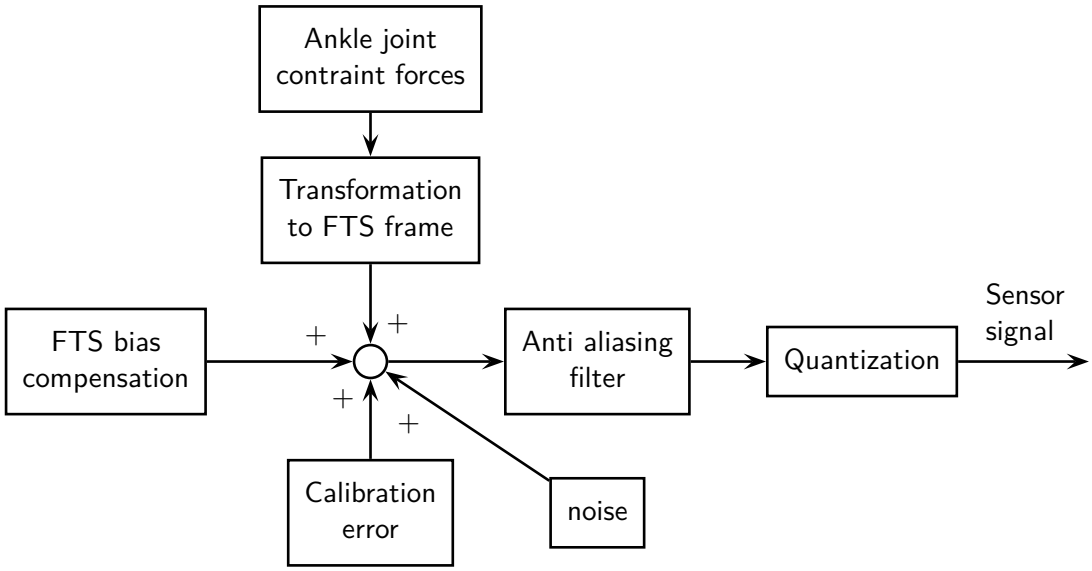


Figure 2.23: Simulation model for force/torque sensors

Johnnie's sensor is a custom development based on MEMS[13] gyroscopes and accelerometers, while a commercial system[14] based on fiber optic gyroscopes (FOG) is used in Lola.

Angular velocity and upper body orientation are directly calculated from the generalized coordinates $\boldsymbol{q}$ during simulations. However, the most relevant measurement errors are taken into account by global error models applied to the resulting upper body orientation and angular velocity signals.

Limited bandwidth is modeled by low-pass filtering the ideal angular velocity and orientation. When initialized correctly, the angular velocity does not show any bias. However, the orientation signal always has a certain constant bias due to imperfect mounting alignment of the IMU. Finally, white noise and signal quantization is taken into account. Figure 2.24 shows the overall IMU model for dynamics simulations.

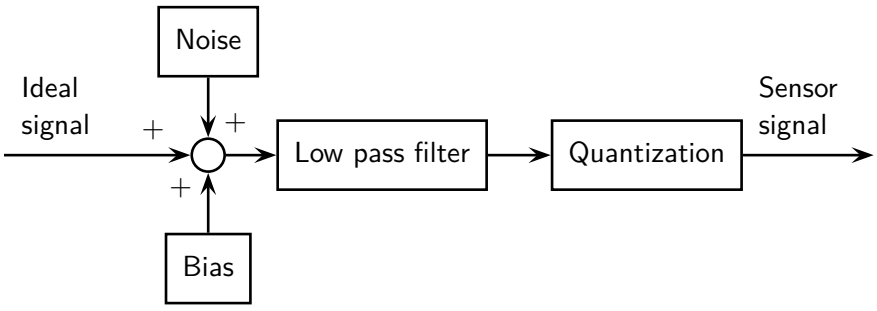


Figure 2.24: Simulation model for the inertial measurement unit

13 Micro-Electro-Mechanical-Systems
14 iVRU-FC-C167-200/2-200-24 from iMAR http://www.imar-navigation.de

2.6 Robot Models and Time Integration

2.6.1 Robot Models

By combining the different models for rigid body mechanics, drive and contact dynamics, a family of simulation models with varying modeling depth is constructed. Table 2.4 and Table 2.5 give an overview of the different models that were implemented for Lola and Johnnie. FE-models were not implemented for Johnnie, since a simple point contact is a good approximation for the small contact elements used in Johnnie's feet (cf Figure 2.10).

For independent spring-damper contact models, one contact at the edge of each contact element is used. The FE contact models consist of 8-node hexahedral elements and have a total of 72 nodes for each of the four contact elements in every foot. To decrease the computational cost, the 36 nodes rigidly attached to the rigid foot segment are eliminated in an off-line preprocessing step, leading to a minimal coordinate representation with 36 nodes, or 108 DoFs, per contact element. Figure 2.25 shows the mesh of one of Lola's contact elements.

Implementing such a large number of models is motivated by the fact that the simulation is used for a wide range of different tasks. The detailed models are used for calculating loads during the design phase of Lola. Here, the required accuracy is high and detailed data, such as the required currents for different motors or actuator torque and speed as function of drive kinematics, should be calculated. Elastic gear models are especially useful when studying the low level joint control loop or fast walking. The models are divided into two groups according to the structure of the resulting EoM, which are described in the following: (1) Full models (type 2 to type 5) and (2) a reduced model (type 1).

Table 2.4: Overview of simulation models for Lola with different modeling depth

	Reduced model	Full models			
Model component	Type 1	Type 2	Type 3	Type 4	Type 5
Rigid body dynamics	yes	yes	yes	yes	yes
Gear elasticity	no	no	yes	no	yes
Drive dynamics	no	yes	yes	yes	yes
Contact model	Independent	Independent	Independent	FEM	FEM
Mechanical DoFs	6	30	46	30	46
Contact layer DoFs	48	48	48	864	864
Electrical DoFs	0	24	24	24	24
No. of 1st order ODEs	60	132	164	948	980

Full Models

The full models comprise rigid multibody dynamics, drive and contact dynamics. Depending on which contact model is used and if gear elasticity is taken into account,

Table 2.5: Overview of simulation models for Johnnie with different modeling depth

	Reduced model	Full models	
Model component	Type 1	Type 2	Type 3
Rigid body dynamics	yes	yes	yes
Gear elasticity	no	no	yes
Drive dynamics	no	yes	yes
Contact model	Independent	Independent	Independent
Mechanical DoFs	6	21	26
Contact layer DoFs	24	24	24
Electrical DoFs	0	15	15
No. of 1st order ODEs	36	81	91

the number of DoFs and some components of the EoM will vary, but the basic structure of the set of equations and constraints remains the same:

$$\begin{aligned} \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{h} &= \boldsymbol{Q}_{\text{drive}} + \boldsymbol{Q}_{\text{cont}} \\ \boldsymbol{L}\dot{\boldsymbol{I}} + \boldsymbol{R}\boldsymbol{I} + \boldsymbol{k}_M\boldsymbol{\omega}_{\text{rot}} &= \boldsymbol{U} \\ \boldsymbol{B}\dot{\boldsymbol{d}} + \boldsymbol{K}\boldsymbol{d} &= \boldsymbol{f} \\ \lambda_{N,i} &= \text{prox}_{\mathscr{C}_N}(\lambda_{N,i} - r\dot{g}_{N,i}) \qquad \forall \quad i \in I_c \\ \boldsymbol{\lambda}_{T,i} &= \mathbf{prox}_{\mathscr{C}_T(\lambda_N)}(\boldsymbol{\lambda}_{T,i} - r\dot{\boldsymbol{g}}_{T,i}) \quad \forall \quad i \in I_c \end{aligned} \tag{2.71}$$

This set of equations can be rewritten as a first order ODE:

$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}) \tag{2.72}$$

$$\boldsymbol{z}^T = (\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T, \boldsymbol{I}, \boldsymbol{d}) \tag{2.73}$$

Calculating $\dot{\boldsymbol{z}}$ implies solving (2.71) for $\ddot{\boldsymbol{q}}, \dot{\boldsymbol{I}}$ and $\dot{\boldsymbol{d}}$. $\dot{\boldsymbol{d}}$ is computed together with $\boldsymbol{f}$ in the contact solver and $\dot{\boldsymbol{I}}$ is numerically cheap to calculate, since $\boldsymbol{L}$ is diagonal. Explicitly solving $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{h} = \boldsymbol{Q}_{\text{ext}}$ for $\ddot{\boldsymbol{q}}$ could be avoided by using one of the well-known recursive $O(n)$ algorithms proposed by Brandl, Hollerbach, Bae and others. An overview of relevant methods is given in [107]. However, since solving for $\ddot{\boldsymbol{q}}$ using a Cholesky factorization of $\boldsymbol{M}$ does not take up a significant percentage of the simulation time[15], such an algorithm was not considered here.

Reduced Model

Simulation times and model complexity can be reduced significantly by taking the robot's control architecture into account. Johnnie and Lola are controlled by a hierarchical walking control system with a high-gain joint position control loop on the lowest level. In deriving the reduced model, the following assumptions are made for the sake of simplicity and without loss of generality:

15 Measurements of a "type 2" simulation with Intel's VTune (http://software.intel.com/en-us/intel-vtune/) profiler showed that $< 1\%$ of the time was spent calculating the Cholesky factorization.

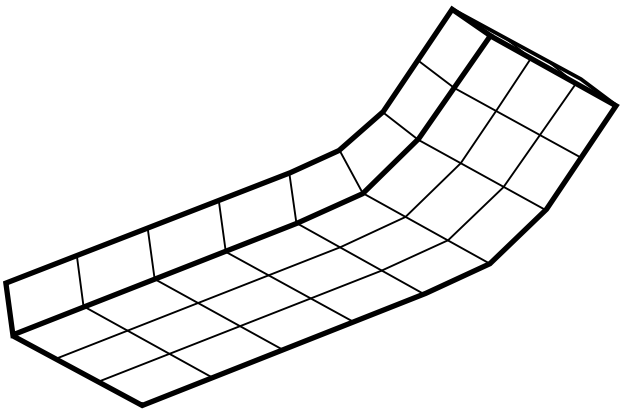


Figure 2.25: Mesh of the FE model for one of Lola's contact elements consisting of 8-node hexahedral elements.

1. A simple proportional control law is used
2. The contact state does not change

Dividing the generalized coordinates into six torso DoFs $\boldsymbol{q}_T$ and $(n-6)$ joint angles $\boldsymbol{q}_J$, the equations of motion can be rewritten as:

$$\boldsymbol{M}_{TT}\ddot{\boldsymbol{q}}_T + \boldsymbol{M}_{TJ}\ddot{\boldsymbol{q}}_J + \boldsymbol{h}_T^\star = \boldsymbol{0} \tag{2.74}$$

$$\boldsymbol{M}_{JT}\ddot{\boldsymbol{q}}_T + \boldsymbol{M}_{JJ}\ddot{\boldsymbol{q}}_J + \boldsymbol{h}_J^\star - K\boldsymbol{\Delta q}_J = \boldsymbol{0} \tag{2.75}$$

$$\boldsymbol{\Delta q}_J = \boldsymbol{q}_J - \boldsymbol{q}_{J,d} \tag{2.76}$$

The vectors $\boldsymbol{h}_i^\star$ include contact forces and joint torques, $\boldsymbol{\Delta q}_J$ is the joint angle tracking error, $\boldsymbol{q}_{J,d}$ are the desired joint angles and K is a control parameter.

Since $K = 1/\epsilon$ is large, a reduced model can be derived using singular perturbation theory [54]. Substituting $\boldsymbol{q}_J = \boldsymbol{q}_{J,d} + \boldsymbol{\Delta q}_J$ and a series approximation for $\boldsymbol{\Delta q}_J$

$$\boldsymbol{\Delta q}_J \approx \boldsymbol{\Delta q}_{J,0} + \epsilon\boldsymbol{\Delta q}_{J,1} \tag{2.77}$$

into Equations (2.74) to (2.76) and rearranging yields:

$$\boldsymbol{M}_{TT}\ddot{\boldsymbol{q}}_T + \boldsymbol{M}_{TJ}(\ddot{\boldsymbol{q}}_{J,d} + \boldsymbol{\Delta}\ddot{\boldsymbol{q}}_{J,0} + \epsilon\boldsymbol{\Delta}\ddot{\boldsymbol{q}}_{J,1}) + \boldsymbol{h}_T^\star = \boldsymbol{0} \tag{2.78}$$

$$\epsilon\boldsymbol{M}_{JT}\ddot{\boldsymbol{q}}_T + \epsilon\boldsymbol{M}_{JJ}(\ddot{\boldsymbol{q}}_{J,d} + \boldsymbol{\Delta}\ddot{\boldsymbol{q}}_{J,0} + \epsilon\boldsymbol{\Delta}\ddot{\boldsymbol{q}}_{J,1}) + \epsilon\boldsymbol{h}_J^\star - \boldsymbol{\Delta q}_{J,0} - \epsilon\boldsymbol{\Delta q}_{J,1} = \boldsymbol{0} \tag{2.79}$$

The vectors $\boldsymbol{\Delta q}_{J,i}$ can be calculated by matching terms with like powers in ϵ. By matching terms with ϵ^0, the *unperturbed solution* for $\epsilon \to 0$ is obtained:

$$\boldsymbol{M}_{TT}\ddot{\boldsymbol{q}}_T + \boldsymbol{M}_{TJ}\ddot{\boldsymbol{q}}_{J,d} + \boldsymbol{h}_T^\star = \boldsymbol{0} \tag{2.80}$$

$$\boldsymbol{\Delta q}_{J,0} = \boldsymbol{0} \tag{2.81}$$

That is, the time evolution of the upper body coordinates can be obtained by integrating the six ODEs (2.80), while the joint angles follow the desired trajectories.

It must be emphasized that (2.80) contains the full multibody dynamics. Also, setting up these equations is no faster than for a full model, since the contributions of all bodies in the system are taken into account. However, since only the motion on the *slow manifold* is determined, at least 10 times larger integrator step sizes can be chosen. In fact, a reduced model for Johnnie and Lola can be simulated in real-time on a standard PC.[16] Therefore, the reduced model is very useful for quickly checking global stability of the walking system. Also, this model enables an analysis of the ideal case of perfect joint trajectory tracking, which is important for systems based on an inner joint position control loop. A first order approximation of the *perturbed solution* could be obtained by comparing terms with ϵ^1.

Partitioning an underactuated mechanical system into actuated ($\boldsymbol{q}_J$) and unactuated ($\boldsymbol{q}_T$) DoFs is a common approach in the context of control system analysis and design. From a control systems point of view, equation (2.80) describes the zero dynamics of the robot with joint position control [112]. Achieving stable zero dynamics is a major problem in biped walking control. Therefore, the reduced model provides an efficient means for determining the performance and stability of trajectory generation and stabilizing control.

2.6.2 Time Integration

In order to calculate the time evolution of the state variables, the equations of motion are rewritten as a standard first order ordinary differential equation:

$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}, \boldsymbol{u}) \tag{2.82}$$

The state vector $\boldsymbol{z}$ and input vector $\boldsymbol{u}$ depend on the type of simulation. For the reduced model

$$\boldsymbol{z}^T = \begin{pmatrix} \boldsymbol{q}_T^T & \dot{\boldsymbol{q}}_T^T & \boldsymbol{d}^T \end{pmatrix} \tag{2.83}$$

$$\boldsymbol{u}^T = \begin{pmatrix} \boldsymbol{q}_J^T, \dot{\boldsymbol{q}}_J^T, \ddot{\boldsymbol{q}}_J^T \end{pmatrix} \tag{2.84}$$

and for the full model

$$\boldsymbol{z}^T = \begin{pmatrix} \boldsymbol{q}^T & \dot{\boldsymbol{q}}^T & \boldsymbol{d}^T & \boldsymbol{I}^T \end{pmatrix} \tag{2.85}$$

$$\boldsymbol{u} = \boldsymbol{U} \tag{2.86}$$

Since contact forces $\boldsymbol{f}$ and deformation rates $\dot{\boldsymbol{d}}$ can be calculated independently of $\boldsymbol{q}, \dot{\boldsymbol{q}}$, the contact constraints can be solved independently from time integration. The result is then used to calculate $\boldsymbol{f}(\boldsymbol{z}, \boldsymbol{u})$. Dry friction and the damping properties of the contact elements make $\boldsymbol{f}(\boldsymbol{z}, \boldsymbol{u})$ discontinuous, while the state variables remain continuous but are non-smooth. In the terminology of non-smooth dynamics, the robot is a Filippov-type system [57].

16 Using one core of an Intel E8400 CPU running at 3.00 GHz, Johnnie's simulation takes 0.8× real-time, while Lola's takes 1.4× real-time. That is, the simulation time for the reduced model is faster than real-time for Johnnie and a little slower for Lola. The simulation time scales almost linearly in the number of bodies in the multibody system.

The basic task of the simulation program is to calculate $\dot{\boldsymbol{z}}$, which is then integrated by an appropriate algorithm to obtain $\boldsymbol{z}(t)$. Figure 2.26 shows the basic program flow for calculating $\dot{\boldsymbol{z}}$.

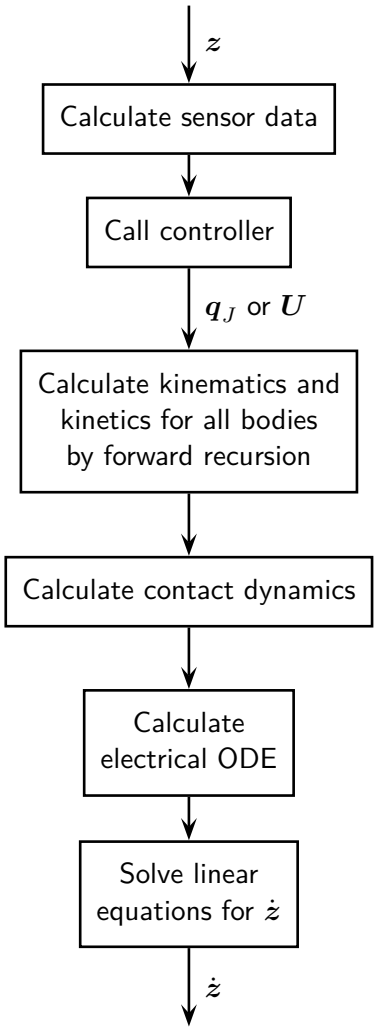


Figure 2.26: Program flow for calculating $\dot{\boldsymbol{z}}$

In general, the choice of a suitable integration algorithm depends on the properties of the differential equation. Higher order integrators for ODEs such as Runge-Kutta, Adams or BDF (Backward Differentiation Formula) methods are well suited for many simulations of mechatronic systems, since highly efficient implementations are readily available that provide very accurate results. However, these methods assume a certain order of smoothness of the ODE. If the algorithm automatically adjusts the step size, such an integrator might calculate a reasonably accurate solution even if the smoothness assumptions are violated. However, this comes at the cost of very small time steps close to discontinuities [1, 57]. Besides using an integrator for smooth ODEs and hoping for the best, there are two main approaches to time-integration for discontinuous ODEs: event-driven methods and time stepping schemes [1].

Event-driven methods are very accurate and a good choice if the exact time of

discontinuities has a significant influence on the system's trajectory. Since this is not the case for the proposed robot models, an event-driven integrator was not implemented.

Instead, both a simple explicit Euler scheme, which does not assume a smooth right hand side, and the Livermore Solver for Ordinary Differential Equations (LSODE) [96] with automatic step size determination were used. For a given accuracy, simulations with LSODE are approximately 1.5 to 2.0 times slower.

2.7 Chapter Summary

This chapter gave an overview of robot models developed for Johnnie and Lola. The simulation system is designed as a library of component models, contact and ordinary differential equation solvers. The model library includes components for rigid multibody dynamics, nonlinear drive mechanisms, nonlinear gear elasticity, nonlinear friction and electrical motor dynamics. The unilateral, viscoelastic contact can be modeled using either independent spring-damper systems or finite element models. Sensor models were presented that include limited bandwidth, noise, quantization and bias. Not all model components must be used in a robot simulation. Rather, only the components required for the current simulation task can be activated. Using this approach, either fast (and less detailed) or slower (but very detailed) simulations can be performed. The chapter also includes the description of a detailed three-dimensional environment model that enables simulating walking over uneven ground or climbing onto or over objects. Finally, numerical aspects of contact solvers and time integration were presented.

3 Stability and Feasibility in Biped Walking

3.1 Introduction

Maintaining stability is of central importance in biped locomotion. Reducing energy consumption is often chosen as an optimization target for legged robots in the hope of achieving "natural" movements closer to those of human beings. Nevertheless, research has shown that stability, not energy consumption, is the most important criterion for humans — at least when avoiding unexpected obstacles [70].

This chapter reviews basic problems in biped walking control, generating feasible motions and maintaining balance, by examining the basic structural properties of biped robot dynamics. Moreover, it gives an overview of concepts often used to study the stability of biped robots.

3.2 Basic Aspects of Biped Walking Dynamics

Classical robot manipulators have bases fixed to the environment and one actuator per joint, i.e., they are *fully actuated*. Control of such robots is a well established field and stability analyses for many standard joint space and task-space control methods have been published (see [111] for an overview). Biped robots are different from such fixed-base manipulators, because they can only transmit forces to the environment via unilateral contacts. These unilateral contacts impose constraints on the *feasible contact forces* depending on the current robot state (cf. Section 2.3.1). Moreover, the system is structure varying: if the contact forces are on the boundary of the set feasible forces, the contact state can change. Such changes regularly occur during normal walking when the stance leg switches. However, they can also be caused by disturbances. Figure 3.1 shows an example where the contact state changes from surface to line contact due to an external disturbance. Such involuntary changes of the contact state can be catastrophic, since the robot looses the ability to control one of the unactuated degrees of freedom.

If the contacts are compliant, as is the case for systems considered here, the robot also is *underactuated*, since there are more degrees of freedom than actuators.

The robot's dynamics can be partitioned into joint angles and upper body degrees of freedom ($\boldsymbol{q}_T \in \mathbb{R}^6$ and $\boldsymbol{q}_J \in \mathbb{R}^{n-6}$, cf. Section 2.6.1). Stability of joint angle trajectories $\boldsymbol{q}_J$ is generally not a problem, since it can easily be achieved using control methods developed for robot manipulators. The challenge is to stabilize the upper body degrees of freedom or, equivalently, to stabilize the total linear ($\boldsymbol{p}$) and angular ($\boldsymbol{L}_{\mathrm{CoG}}$) momentum of the robot:

$$\dot{\boldsymbol{p}} = m\boldsymbol{g} + \sum_{i=1}^{2} \boldsymbol{F}_i \tag{3.1}$$

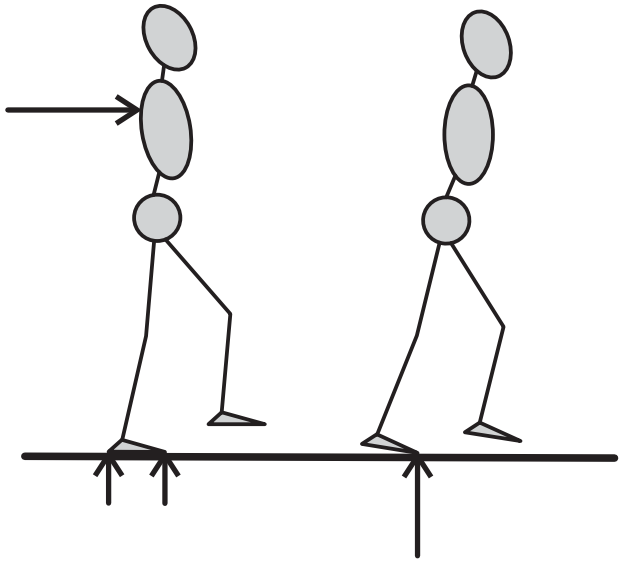


Figure 3.1: Disturbances can change the contact state of a biped

$$\dot{\boldsymbol{L}}_{\mathrm{CoG}} = \sum_{i=1}^{2} (\boldsymbol{T}_i - \boldsymbol{r}_{\mathrm{CoG},\boldsymbol{F}_i} \times \boldsymbol{F}_i) \tag{3.2}$$

$$\dot{\boldsymbol{p}} = m\ddot{\boldsymbol{r}}_{\mathrm{CoG}} \tag{3.3}$$

$${}_I\dot{\boldsymbol{L}}_{\mathrm{CoG}} = \frac{d}{dt} \sum_{k=1}^{n_{\mathrm{bodies}}} \left[m({}^k_I\boldsymbol{r}_o - {}^k_I\boldsymbol{r}_{\mathrm{CoG}}) \times {}^k_I\dot{\boldsymbol{r}}_{\mathrm{CoG}} + {}^k\boldsymbol{A}_{Ik}\, {}^k_k\boldsymbol{I}_o\, {}^k_k\boldsymbol{\omega} \right] \tag{3.4}$$

Here $\boldsymbol{r}_{\mathrm{CoG},\boldsymbol{F}_i}$ denotes the vector from the CoG to the i-th external force $\boldsymbol{F}_i$ and $\boldsymbol{F}_i, \boldsymbol{T}_i$ are the forces acting on the i-th foot. The time evolution of $\boldsymbol{p}$ and $\boldsymbol{L}_{\mathrm{CoG}}$ is determined by the contact forces, which in turn can be controlled via the joint angles or joint torques.

3.3 Zero Moment Point and Related Concepts

3.3.1 Zero Moment Point

A widely used concept in the area of biped robots is the so-called zero moment point, or ZMP, proposed by Vukobratović more than 30 years ago [129]. The concept of the ZMP is only valid for a robot walking on a flat surface with *sufficient friction*, i.e., it is assumed that it cannot slip [128].

All external forces acting on the robot can be summed up to the total contact force $\boldsymbol{F}_c$ and contact moment $\boldsymbol{T}_c^p$. The numerical values of $\boldsymbol{T}_c^p$ depend on the position of the chosen reference point p. The ZMP is defined as the reference point in the contact plane, for which the horizontal components $T_{c,x}, T_{c,y}$ of the contact moment vanish. This is equivalent to the center of pressure (CoP) widely used in biomechanics [27, 106].

If $\boldsymbol{F}, \boldsymbol{T}$ are known for some reference point o, the position of the ZMP is computed

from

$$\boldsymbol{T}_c^{\text{zmp}} = \boldsymbol{T}_c^o - \boldsymbol{p} \times \boldsymbol{F} \tag{3.5}$$

which can be solved for p_x, p_y:

$$\begin{aligned} p_x &= -\frac{T_{c,y}}{F_{c,z}} \\ p_y &= \frac{T_{c,x}}{F_{c,z}} \end{aligned} \tag{3.6}$$

The ZMP concept states that the robot is *dynamically balanced* if the ZMP is within the robot's support polygon, which is defined as the convex hull of all contact points. Moreover, the distance from ZMP to the boundary of the support polygon is often used as a stability measure or margin of stability [28, 36]. There is a clear analogy between the ZMP concept and stability in the static case. In the static case the projection of the center of gravity must lie within the convex hull of all contact points, if the system is to remain at rest. While the term "dynamically balanced" is acceptable (as will be explained later), some authors use stronger terms that, strictly speaking, are incorrect. The following statement is representative: "If the ZMP is inside the contact polygon between the foot and the ground, the biped robot is stable" [36]. Contrary to this statement, the ZMP is not directly related to stability. The ZMP is inside the support polygon *for every physically feasible motion*, as will be shown in the following (see Section 3.3.4). Note, however, that the original wording used by Vukobratović is correct, since the ZMP being inside the support polygon is in fact equivalent to the robot being "dynamically balanced," since the laws of classical mechanics require the forces acting on the robot to be "balanced." That is, the ZMP concept provides a test if the motion of a biped is consistent with the basic laws of mechanics.

To clarify the meaning of the ZMP, its relationship to the unilateral foot-ground contact will be highlighted in the following. Denoting the contact forces acting on the robot as $\boldsymbol{f}_i$ and assuming there are only point contacts, the total ground reaction force and moment are given by:

$$\boldsymbol{F}_c = \sum_{i=1}^{n_c} \boldsymbol{f}_i \tag{3.7}$$

$$\boldsymbol{T}_c^o = \sum_{i=1}^{n_c} \boldsymbol{r}_{oi} \times \boldsymbol{f}_i \tag{3.8}$$

where n_c are the number of contact points and $\boldsymbol{r}_{oi}$ is the vector from the reference point o to the i-th contact point. Since the contact is unilateral, the vertical force components must be positive:

$$f_{z,i} \geq 0 \qquad \forall i \in \{1 \dots n_c\} \tag{3.9}$$

Using (3.8), the horizontal moments can be written as:

$$T^o_{c,x} = F_{c,z} \sum_{i=1}^{n_c} \frac{f_{i,z}}{F_{c,z}} r_{oi,y} \tag{3.10}$$

$$T^o_{c,y} = F_{c,z} \sum_{i=1}^{n_c} -\frac{f_{i,z}}{F_{c,z}} r_{oi,x} \tag{3.11}$$

Substituting $\alpha_i = \frac{f_{i,z}}{F_{c,Z}}$ and (3.6) into these equations and rearranging yields:

$$p_y = \sum_{i=1}^{n_c} \alpha_i r_{oi,y} \tag{3.12}$$

$$p_x = \sum_{i=1}^{n_c} \alpha_i r_{oi,x} \tag{3.13}$$

$$\alpha_i \geq 0 \quad \forall i \qquad \wedge \qquad \sum_{i=1}^{n_c} \alpha_i = 1 \tag{3.14}$$

Comparing these equations to the definition of the convex hull $\mathrm{co}(\boldsymbol{P})$ of a set of points $\boldsymbol{P}$

$$\mathrm{co}(\boldsymbol{P}) = \left\{ \sum_i \alpha_i \boldsymbol{p}_i \,\middle|\, \boldsymbol{p}_i \in \boldsymbol{P}, \quad \alpha_i \geq 0, \quad \sum_i \alpha_i = 1 \right\} \tag{3.15}$$

shows that the ZMP criterion is equivalent to the original unilateral constraints for the ground contact.

Nevertheless, the ZMP has become widely used since it is easily understood and appears to offer a straightforward extension from static to dynamic stability. Also, the ZMP is widely used in algorithms for generating reference trajectories. Here the ZMP criterion is indeed useful for generating feasible trajectories that can be executed by a physical robot. Moreover, the distance of the ZMP from the boundary of the support polygon is significant, if the robot's posture is controlled via the contact moments or the ZMP. In this case, the robot loses its ability to stabilize the posture in one direction if the ZMP is on the boundary of the support polygon.

3.3.2 Foot Rotation Indicator and Zero Rate of Change of Angular Momentum

Goswami presented the concept of the Foot-Rotation Indicator (FRI) as an extension of the ZMP concept for a biped robot in single support [27]. The FRI is defined as the point in the contact plane where the ground reaction force would have to act in order to prevent the robot's stance foot from rotating. When the foot is at rest, the FRI is equal to the ZMP. When the foot does rotate about its edge, the FRI is outside of the support polygon and the distance to the edge is proportional to the moment producing the foot rotation.

More recently Goswami proposed the Zero Rate of Change of Angular Momentum (ZRAM) as a generalization of the FRI to the case of multiple, non-coplanar contacts

[28]. In the cited paper it is assumed that a biped is stable, if the angular momentum about the center of gravity $\boldsymbol{L}_{\mathrm{CoG}}$ is constant and that it tends to tip over otherwise. When $\dot{\boldsymbol{L}}_{\mathrm{CoG}} \neq 0$, the ZRAM is defined as the point where the ground reaction force would have to act, in order to achieve $\dot{\boldsymbol{L}}_{\mathrm{CoG}} = 0$.

While the angular momentum will change if the robot falls, it is not necessarily constant for stable gait patterns. In fact, this criterion is violated for the gaits of Johnnie and Lola. In the author's experience, gaits satisfying this criterion appear very unnatural and lead to large compensating motions of arms and/or the upper body in order to cancel the change of angular momentum produced by the legs. This observation is confirmed by [116], where the angular momentum is not set to zero to avoid "unusual" movements.

3.3.3 Stability Criteria Based on the Contact Wrench

Hirukawa et al. [33] propose a "universal stability criterion" based on the contact wrench acting on the robot's feet. This concept exploits the fact that the contact force at each point contact must lie within the friction cone. In addition, the total contact wrench must also lie within a convex cone defined by the location of the point contacts and the coefficient of friction. In their derivation, Hirukawa et al. used a linearized friction cone to obtain an equation for the polyhedral convex cone of feasible contact wrenches. This criterion allows non-coplanar contacts and can therefore be used in the general case of a humanoid in contact with the environment. In the case of coplanar contacts, it is equivalent to the ZMP. Therefore, this criterion has the same relevance to biped stability as the ZMP.

In a related approach, Takao previously proposed the "Feasible Solution of Wrench" (FSW) [118]. However, the author does not give a mathematical representation of the set of feasible contact wrenches and does not treat the relationship of the FSW to contact stability.

3.3.4 Remarks

Except for FRI and ZRAM, the "stability criteria" presented above are equivalent to the original inequality constraints on the contact forces due to the unilateral contact with the environment. In fact, "feasibility" would be a better term than "stability," since satisfying these criteria does not guarantee stability.

Figure 3.2 shows two robot states for which the center of gravity x and ZMP position p_x are both zero, but the center of gravity velocity $\dot{x}$ differs.

To analyze the stability of these two configurations, the robot is modeled as an inverted pendulum (cf. Section 4.6):

$$\ddot{x} = \frac{g}{h}(x - p_x) = \alpha(x - p_x) \tag{3.16}$$

Here h is the height of the center of gravity. For the sake of simplicity, it is assumed that the robot is in a standing position and p_x is constant. The system trajectory is

then given by:

$$x(t) = p_x + \frac{e^{\sqrt{\alpha}t}}{2\sqrt{\alpha}} \underbrace{\left(\dot{x}_0 + \sqrt{\alpha}(x_0 - p_x)\right)}_{A} + \frac{e^{-\sqrt{\alpha}t}}{2\sqrt{\alpha}} \underbrace{\left(-\dot{x}_0 + \sqrt{\alpha}(x_0 - p_x)\right)}_{B} \tag{3.17}$$

Here $(x_0, \dot{x}_0)$ is the system state at $t = 0$. The trajectory diverges if A cannot be made to zero, i.e., if

$$p_x^\star = x_0 + \frac{\dot{x}_0}{\sqrt{\alpha}} \tag{3.18}$$

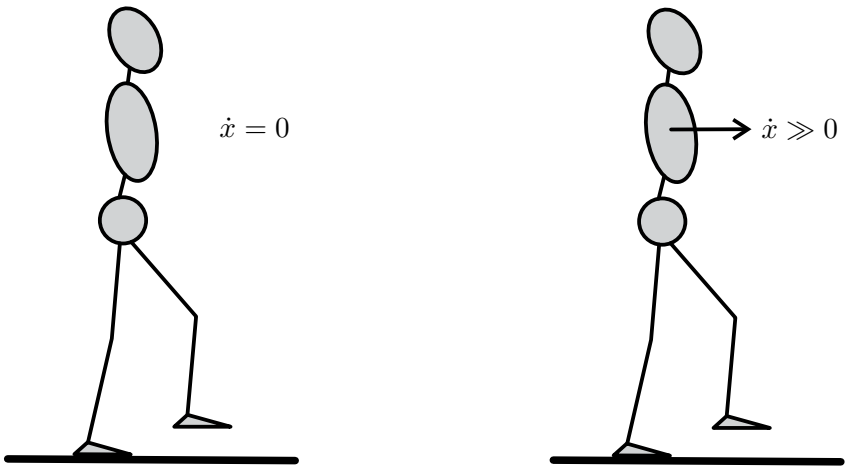
is not within the support polygon. That is, the robot can be stabilized, if p_x can be shifted to $p_x^\star$ by a controller, which is impossible for $\dot{x}_0 \gg 0$, irrespective of the current ZMP and CoG positions.

This example shows that, while useful in some cases, the widely used "stability criteria" are not suitable for determining stability or instability in biped locomotion.

3.4 General Stability Criteria

Lyapunov Stability

In the context of nonlinear systems, stability of equilibrium points and orbital stability are usually specified in the sense of Lyapunov [63]. Lyapunov stability theorems [86, 135] are applicable to highly complex systems and have often been used for proving stability of control laws for robot manipulators. At the same time, the original theorems due to Lyapunov are only applicable to smooth systems, i.e., not to biped robots with changing ground contacts. While there are extensions to the non-smooth case [11, 31, 109], these have (to the author's knowledge) not been used in the practical synthesis or analysis of walking controllers for biped robots.



(a) Stable (b) Unstable

Figure 3.2: For given ZMP and CoG positions, a biped can be either (a) stable or (b) unstable.

Viability Kernel

To rigorously define when a walking system is stable, Wieber [131] proposed the concept of a *viability kernel*. Defining $\mathcal{F}$ as the set of all system states in which the robot has fallen, the robot is considered to be stable at $t = t_0$, if:

$$\boldsymbol{x}(t) \notin \mathcal{F} \quad \forall\, t \geq t_0 \tag{3.19}$$

The union of all states satisfying (3.19) is called the viability kernel. The distance of $\boldsymbol{x}$ to the closest non-viable state, called the viability margin, is used as a measure of stability. Wieber proposes computing the viability margin for a given control law using numerical optimization methods. Simply put, this concept states that a robot is unstable if it will fall.

While this definition is general and correct, it is unclear how it could be used to solve the problem of synthesizing a better walking controller. Pragmatically, determining whether a system state is stable or not for a given control law can also be achieved by simply simulating the forward dynamics. A pragmatic approach to determining a "stability margin", which was followed in this research project, is to intentionally introduce measurement errors, modeling errors and disturbances in the dynamics simulation.

3.5 Periodic Motions

By considering only periodic motions, which is reasonable in the context of walking machines, the analysis can be greatly simplified. For a T-periodic orbit of the system, the state $\boldsymbol{x}$ repeats every T seconds:

$$\boldsymbol{x}(t + kT) = \boldsymbol{x}(t), \quad \forall\, k \in \mathbb{N} \tag{3.20}$$

Periodic motion is classically studied using Poincaré maps, which relate the system state at $t = t_0 + T$ to that at $t = t_0$. That is, T-periodic orbits of the original system are transformed into fixed points of the Poincaré map.

For smooth systems, the stability of periodic orbits is determined by the eigenvalues λ_i of the Jacobian matrix of the Poincaré map, the so-called Floquet multipliers. If $\|\lambda_i\| < 1\, \forall i$, the trajectory is stable. It must be emphasized that this analysis is only valid for smooth systems and cannot take into account instabilities due to early foot-ground contact [67].

Sangho used the theory of the stability of period motions to stabilize a planar biped on a periodic orbit [38]. Mombaur used a two-stage optimization scheme to calculate stable periodic orbits for a biped robot model with elastic elements by minimizing the Floquet multipliers [66, 67]. While this method has not been used to generate trajectories for a physical robot, it is a very interesting and theoretically sound approach to this problem. As with all off-line methods for trajectory generation, however, this approach suffers from the "curse of dimensionality." That is, a very large number of trajectories must be generated off-line, if the robot is to vary its step length and step frequency — and the size of the trajectory database increases drastically with every new walking parameter (cf. Section 4.6).

3.6 Chapter Summary

This chapter reviewed basic aspects of biped walking dynamics and gave an overview of widely used stability concepts. The term "stability" is often used imprecisely. Most concepts associated with the term stability, such as the zero moment point (ZMP), are in fact feasibility criteria, i.e., conditions that must hold for all physically feasible motions. Consequently, they are useful in generating feasible reference trajectories, but are not directly related to stability. More general and theoretically sound definitions of stability can and have been given. However, the complex, structure-varying nature of biped walking dynamics makes it difficult to use these concepts to assess the stability of biped robots or synthesize better walking controllers.

4 Real-Time Trajectory Generation

4.1 Introduction

This chapter describes the real-time trajectory generation system developed for this thesis. The basic objective is to generate a stable walking motion from simple, high-level commands such as the desired walking direction and speed. These commands are either input by an operator using a graphical user interface (GUI) or joystick, or obtained from a computer vision system.

Theoretically, such trajectories could be calculated by solving an optimal control problem for a comprehensive robot model. Because of the high computational cost this is not an option for real-time control and autonomous locomotion. The author therefore proposes a hierarchical control system that divides the control problem into smaller tasks that can be solved in real-time. A similar structure for real-time walking control was also proposed by Nishiwaki for the robot H7 [75].

A model-based approach is adopted that is based on the robot models described in Chapter 2 and the stability analysis in Chapter 3. The basic approach is similar to that found in other large, position-controlled robots such as Asimo and HRP-2: the controller (1) calculates ideal reference trajectories that are (2) modified in order to stabilize the robot. The feedback control system is described in Chapter 5. Figure 4.1 illustrates the structure of the walking control system. Note that a strict separation of trajectory planning and feedback control is not always possible, since sensor data is used in some planning algorithms, e.g., in the generation of reference trajectories for the camera head.

4.2 Gait Coordination

This section describes the basics of gait coordination. Since we aim for a human-like gait, the basic structure of the gait cycle, as well as the terminology is taken from biomechanics. Because walking is a fundamentally cyclic process, several recurring phases of motion can be identified. Segmentation of the walking motion into phases can occur at different levels, leading to a larger or smaller number of phases.

The basic unit of gait is a stride, which is equivalent to an entire gait cycle: it starts with initial contact of one foot and ends when the same foot contacts the ground the next time. A stride consists of two steps, divided by initial contact of the second foot. The distance traveled by a foot during one gait cycle is defined as the stride length, while the step length is the distance between two feet during double support (cf. Figure 4.2).

A step can be further partitioned into eight phases: initial contact (IC), loading response (LR), mid stance (MS), terminal stance (TS), pre-swing (PSW), initial swing (ISW) and terminal swing (TSW). The first five phases are grouped into the stance

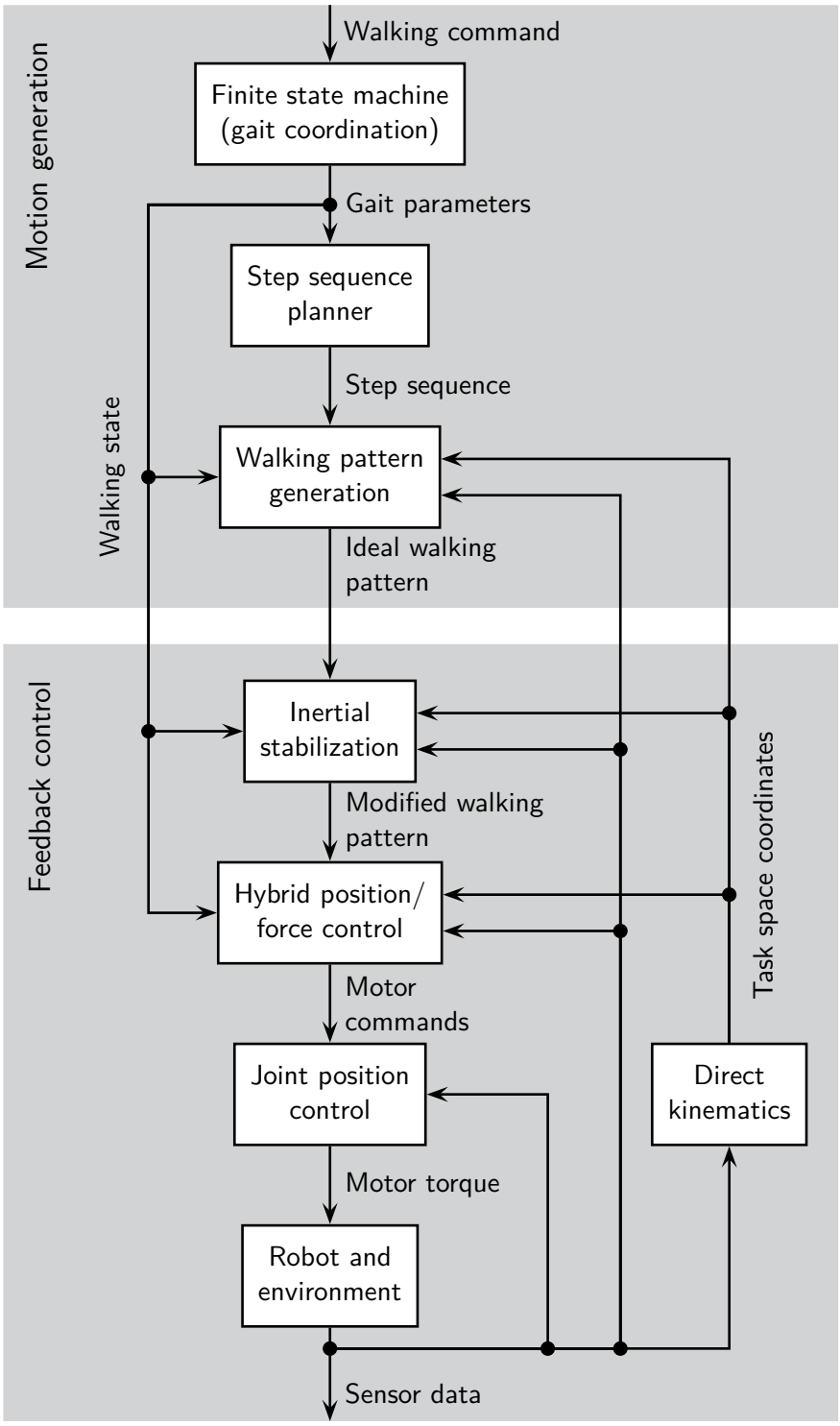


Figure 4.1: Overview of the walking control system

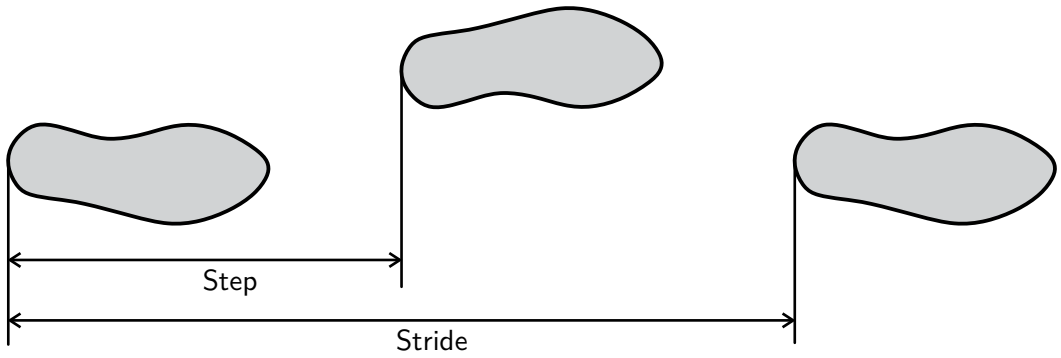


Figure 4.2: Step and stride: a step continues until the next initial contact of the other foot, a stride until the same foot touches the ground again [88].

phase, while the last three are part of the swing phase. Figure 4.3 and Figure 4.4 illustrate these phases. The gait cycles of the right and left leg are synchronized at initial contact, when loading response of the ipsilateral and pre-swing of the contralateral leg begin. In the following, the duration of the i-th phase relative to the step length T_{step} is denoted by p_i, e.g., p_{LR} is the duration of the loading response phase.

The full gait cycle is not always executed, since the robot can take an odd number of steps. Therefore, the step, instead of the stride, is chosen as basic unit of gait in the walking controller.

The basic structure of the gait cycle described above is valid for most walking situations. However, the duration of gait phases and the boundary conditions for trajectory generation are different for periodic, starting and stopping steps. Therefore, the robot motion is further partitioned into standing, starting, periodic walking and stopping phases — that is, a further level is added to the hierarchical representation of walking given above.

Each phase corresponds to a discrete state the robot can be in. Accordingly, the division of walking into a hierarchy of phases directly translates into a description of walking as a hierarchical finite state machine. Figure 4.5 shows the top level of this state machine. State transitions are always triggered by initial contact of the swing leg. Transitions can be triggered either by a sensed ground contact or by a timer indicating the end of the current step. The target of the state transition is determined by the walking commands previously sent to the robot. The current state of the finite state machine determines which parameters are set and which functions are called in lower levels of the control system hierarchy. That is, trajectory planning and stabilizing control are coordinated and configured by the finite state machine.

4.3 Step Sequence Planning

The first level below the finite state machine in the walking controller is a step sequence planner. Depending on the current state and the desired walking parameters, this module calculates a sequence of n_{plan} walking steps.

A step $\mathcal{S}$ is defined by its step parameters $\mathcal{P}$ and augmented by the position and rotation of the reference foot in the current planning frame I, which is located below

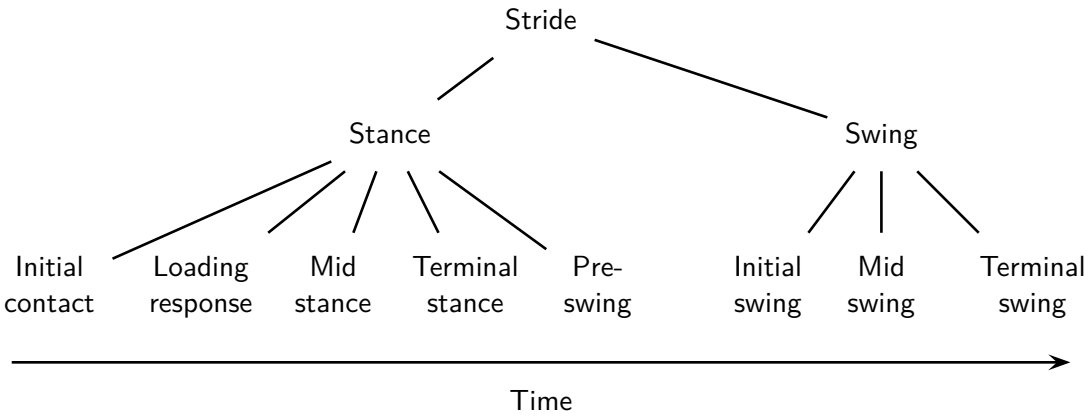


Figure 4.3: Division of the human gait cycle [88]

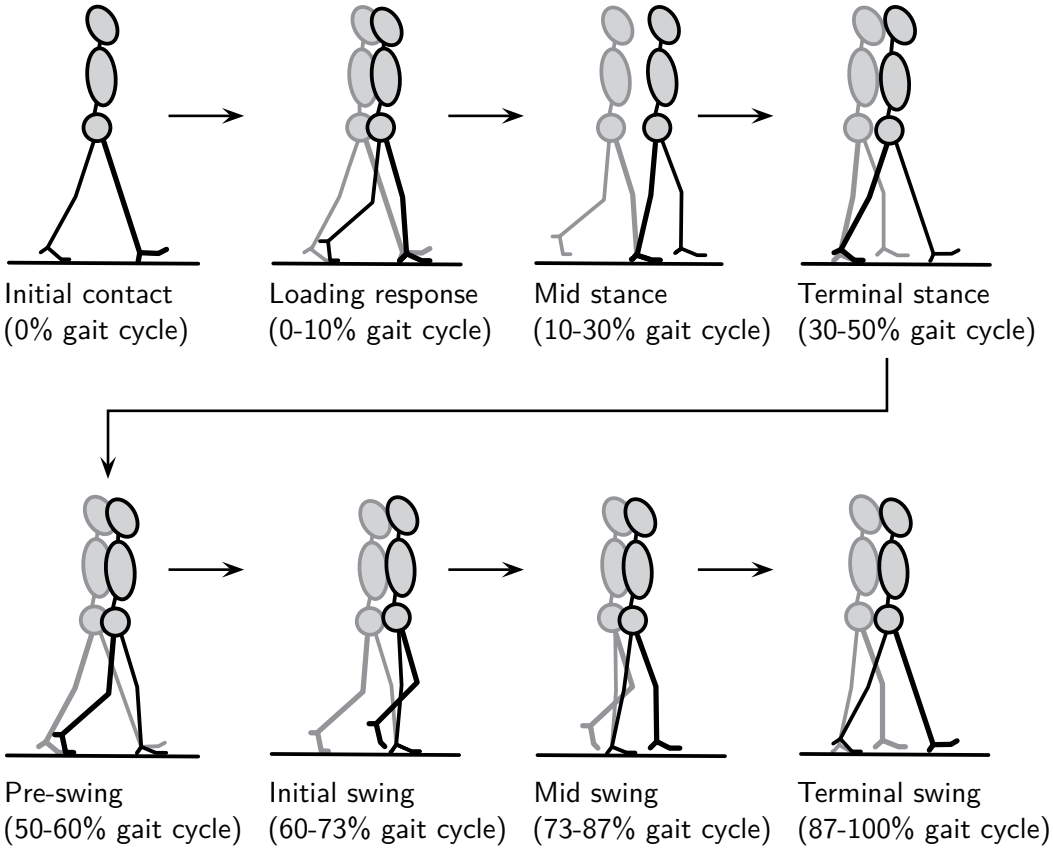


Figure 4.4: Phases of the human gait cycle. Reference limb drawn with thick lines and the pose at the beginning of each phase in gray. Adapted from [88].

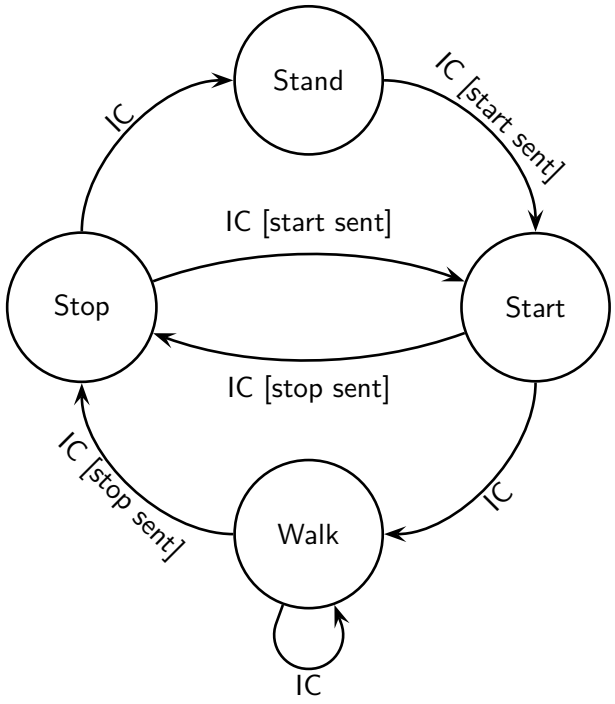


Figure 4.5: Finite state machine for gait coordination. Transitions are triggered by initial contact (IC). Conditions for state transitions shown in brackets.

the current stance foot (see Section 4.4.1 for coordinate system definitions). The most important parameters are listed in Table 4.1. Reference parameters $\mathcal{P}_{\text{ref}}$ for the step sequence are either set via a user interface or calculated from the input obtained from the vision system (cf. Chapter 6). $\mathcal{S}_i$ for the i-th step is calculated from $\mathcal{P}_{\text{ref}}$ and $\mathcal{S}_j$ for the previous steps. Calculation of a step sequence $\mathcal{S}_i, i = \{0 \dots n_{\text{plan}} - 1\}$ from reference parameters $\mathcal{P}_{\text{ref}}$ and obstacle locations is described in the following sections.

4.3.1 Standard Circular Path

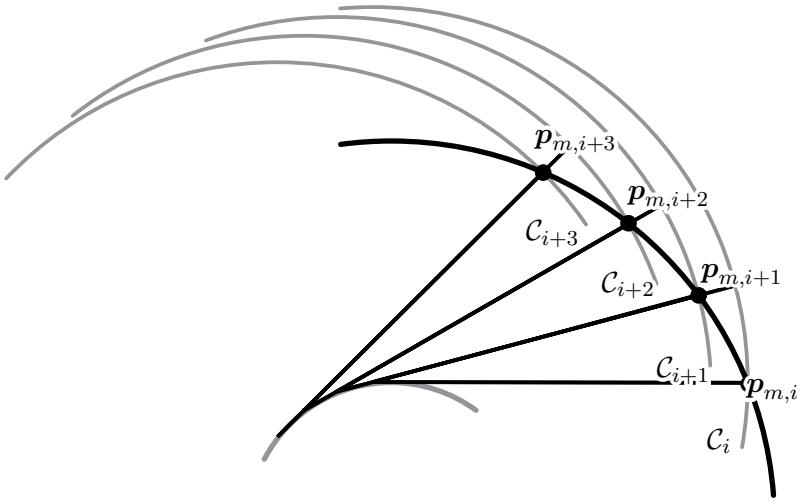
The central part of the step sequence planner is the calculation of footstep locations. The "standard circular path" described in the following enables a simple and intuitive definition of simple paths such as straight lines or circles by choosing just three walking parameters: step length in sagittal (L_x) and lateral direction (L_y) and turning angle φ_{step}. In order to obtain a unified representation, a straight line is represented as a circle with infinite radius.

If $\mathcal{P}_{\text{ref}}$ is constant, the gait is perfectly periodic and the robot follows a circular reference path. Footstep locations of the two feet lie on two concentric circles separated by the step width W_{step}. Figure 4.6(a) illustrates the geometric relationships for four steps. For the sake of clarity, only the reference path between the footsteps is shown. The footsteps are located at a distance of $\pm W_{\text{step}}/2$ from the points $\boldsymbol{p}_{m,k}$ on the reference path, either towards or away from the center of the reference circle $\mathcal{C}_i$.

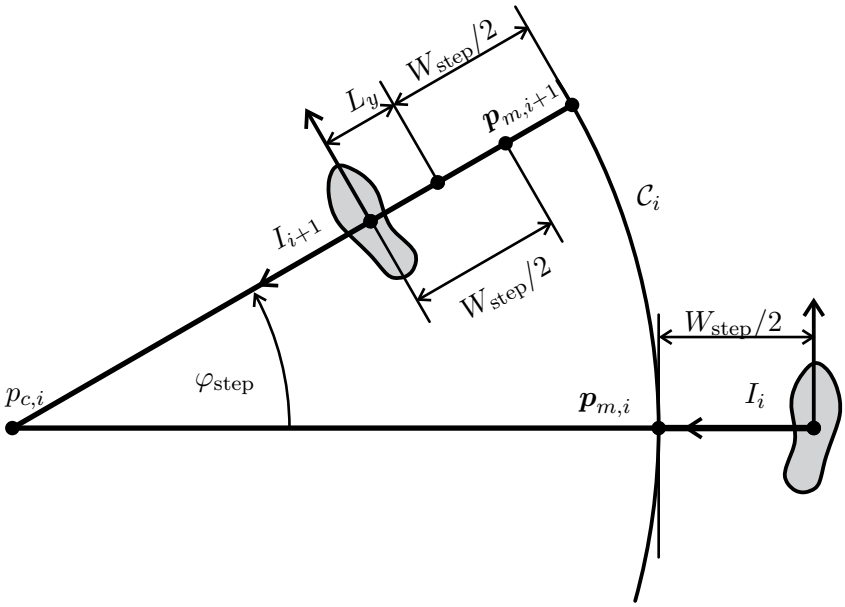
For $L_y = 0$, the robot walks along a circular path, always looking along the tangent

Table 4.1: Important step parameters $\mathcal{P}$

L_x	Step length in sagittal plane
L_y	Step length in lateral plane
W_{step}	Step width
φ_{step}	Relative rotation between feet during double support
z_{swing}	Reference height for swing foot trajectory
T_{step}	Duration of one step
$\boldsymbol{p}_{\text{step}}$	Walking phase durations
$i_s \in \{0, 1\}$	Index of the stance leg (0 : right leg, 1 : left leg)
$\nu_s \in \{-1, 1\}$	Sign function (1 : for $i_s = 0$, -1 : for $i_s = 1$)



(a) Geometry of periodic walking along a circular path



(b) Geometry of one step along a circular path

Figure 4.6: Geometric relationships for the standard circular path

line. In this case, the robot's path and all reference circles coincide. For $L_y \neq 0$ and $L_x = 0$, the robot also walks along a circular path, but faces either towards or away from the center of the circle, depending on the sign of L_y and φ_{step}. If both L_x and L_y are zero, but $\varphi_{\text{step}} \neq 0$, the robot turns around on the spot. When $\varphi_{\text{step}} = 0$, the circle's radius approaches infinity and the robot walks along a straight line.

Figure 4.6(b) illustrates the geometry of a single step in more detail. The next footstep location relative to the current stance foot $\boldsymbol{\Delta r}_{\text{step},i}$ is given by:

$$\begin{aligned} {}_i\boldsymbol{\Delta r}_{\text{step},i} &= {}_i\boldsymbol{p}_{c,i} + \boldsymbol{A}_{i,i+1}\, {}_{i+1}\boldsymbol{r}_{o,i+1} \\ &= \begin{pmatrix} -\sin\varphi_{\text{step}} \left(L_y - \frac{L_x}{\varphi_{\text{step}}} + \frac{W_{\text{step}}\nu_i}{2} \right) \\ \frac{L_x}{\varphi_{\text{step}}} + \frac{W_{\text{step}}\nu_i}{2} + \cos\varphi_{\text{step}} \left(L_y - \frac{L_x}{\varphi_{\text{step}}} + \frac{W_{\text{step}}\nu_i}{2} \right) \end{pmatrix} \end{aligned} \tag{4.1}$$

${}_i\boldsymbol{p}_{c,i}$ is the center of the i-th reference circle $\mathcal{C}_i$, $\boldsymbol{A}_{i,i+1}$ is the rotation matrix transforming vectors from the next step's reference frame I_{i+1} into this step's frame I_i, $\nu_i = \pm 1$ (cf. Table 4.1) and ${}_{i+1}\boldsymbol{r}_{o,i+1}$ is the vector from the center of $\mathcal{C}_i$ to the next stance foot position.

Due to the division by φ_{step}, (4.1) cannot be used to calculate ${}_i\boldsymbol{\Delta r}_{\text{step},i}$ for small turning angles. However, we can obtain a good approximation by using the Taylor series expansion of (4.1):

$${}_i\boldsymbol{\Delta r}_{\text{step},i} \approx \begin{pmatrix} L_x - \left(L_y + \frac{W_{\text{step}}\nu_i}{2} \right) \varphi_{\text{step}} - \frac{1}{6} L_x \varphi_{\text{step}}^2 \\ W_{\text{step}}\nu_i + L_y + \frac{1}{2} L_x \varphi_{\text{step}} - \left(L_y + \frac{W_{\text{step}}\nu_i}{2} \right) \varphi_{\text{step}}^2 \end{pmatrix} \tag{4.2}$$

In the controller implementation, (4.1) is used for $\|\varphi_{\text{step}}\| \geq 0.1\,\text{rad}$ and (4.2) is used otherwise. Even for very large steps with $L_x = 0.7\,\text{m}$, $L_y = 0.2\,\text{m}$, the approximation error is below $10^{-4}\,\text{m}$. Equations (4.1) and (4.2) are applied recursively to calculate footstep locations for all steps $\mathcal{S}_i$.

Note that since a new circular path is calculated for each step, the robot is not restricted to walking along circular paths. Rather, any sequence of footsteps can be generated by changing L_x, L_y and φ_{step} for each step.

4.3.2 Step Parameter Calculation

For calculating $\mathcal{S}_i$, $\mathcal{P}_{\text{ref}}$ is not used directly. Instead, it is modified for each step in order to avoid self-collision and obtain the desired walking behavior.

The first modification concerns the relative rotation between two footsteps. The rules for calculating a step sequence rotate the i-th footstep by $i\varphi_{\text{step}}$. This rotation is performed about the reference frame I_i (cf. Section 4.4.1), which does not necessarily coincide with the center of the sole. To achieve a more natural looking rotation about the center of the foot, the parameters L_x and L_y are modified so that the foot's center $\boldsymbol{p}_{fc}$ is unchanged:

$$\begin{pmatrix} L_x \\ L_y \end{pmatrix}_i = \begin{pmatrix} L_{x,\text{ref}} \\ L_{y,\text{ref}} \end{pmatrix}_i + \boldsymbol{p}_{fc} - \boldsymbol{A}_{i,i+1}\boldsymbol{p}_{fc} \tag{4.3}$$

Simply using (4.1) to calculate the step sequence can lead to overlapping footstep

locations, i.e., self-collisions that would damage the robot. Theoretically, self-collisions can be avoided by many different combinations of foot rotations and translations. However, changes that do not modify the circular reference path defined by $\mathcal{P}_{\text{ref}}$ are preferred. By only changing the step width W_{step}, the reference path itself remains unchanged, only the distance of the footsteps normal to the reference circle changes.

If the minimum acceptable distance between both feet for $L_x = L_y = \varphi_{\text{step}} = 0$ is denoted by $W_{\text{step,min}}$, the minimum distance for $\varphi_{\text{step}} \neq 0$ is given by:

$$W^{\star}_{\text{step,min}} = W_{\text{step,min}} + \left\| \boldsymbol{e}_y^T \left(\boldsymbol{d}_f - \boldsymbol{A}_{i,i+1} \boldsymbol{d}_f \right) \right\| \tag{4.4}$$

Here $\boldsymbol{d}_f$ denotes the vector along the diagonal of the foot sole. The effective step width W_{step} is then obtained by limiting $W_{\text{step,ref}}$ to $W^{\star}_{\text{step,min}}$. Equation (4.4) is a worst case approximation that reliably avoids self-collision.

4.3.3 Reactive Step Sequence Planning

Introduction and Related Work

The general problem of navigation among obstacles is not part of this thesis. Rather, this task is performed by a computer vision system developed in a separate project. Currently, this system defines the desired path in 2-D using circular curve fragments. Obstacles are avoided by choosing a path around them (cf. Chapter 6). A major advantage of bipeds over wheeled robots is their ability to step over small objects and climb stairs. The author proposes planning 2-D paths over small obstacles instead of around them. The geometry and location of obstacles is sent to the robot along with the desired circular reference path, allowing the robot to step over obstacles based on a local, reactive step sequence planning algorithm. In this thesis, obstacles are represented by a "triangle soup," i.e., an unordered list of triangles tessellating the obstacle surface. Related methods for step sequence planning were previously proposed by Cupec [17] and Okada [81].

It must be noted that complex environments with many obstacles may require global planning of footstep locations, since local planning methods can get trapped in local minima. However, if the 2-D reference path is planned globally, local minima for the step sequence planning only exist for very complex environments with few acceptable footholds. Chestnutt et al. proposed a very general method based on $A^{\star}$ search to globally plan a step sequence towards a goal position [12]. This method is capable of finding viable footstep locations in very complex environments. However, the computational cost is significantly higher. Also, the method inherently requires a sampling of feasible footstep locations, artificially limiting the robot's actions.

Proposed Method

In the following, the proposed method for local step sequence planning is described. It is assumed that the 2-D reference path minimizes the step length required for stepping over the obstacle, i.e., the local step sequence planner does not have to modify the reference path itself. Furthermore, the environment is assumed to be "lightly cluttered," allowing the robot to find viable footstep locations using a local

strategy.

Algorithm 3 Collision avoidance for small obstacles

```
1: Plan ideal circular reference path
2: Calculate bounding box of all footstep locations
3: if Obstacle in bounding box then
4:     Find first step with obstacle collision
5:     Adapt step length until obstacle
6:     Plan step over obstacle
7:     Plan remaining steps along ideal reference path
8: end if
```

The method is outlined in Algorithm 3. First, footstep locations are planned without taking the obstacle into account and the first step with an obstacle collision is determined. Steps before the collision are modified, such that the last foothold has a pre-defined minimum distance to the obstacle. To avoid collisions with the obstacle, height and position of the next two steps over the obstacle are planned using the obstacle's bounding box.

Figure 4.7 shows a sequence of snapshots from a dynamics simulation of Lola. In the top row, the robot adapts the footstep locations and the swing foot trajectory in order to step over the obstacle. The bottom row shows results from an animation with the same walking commands, but without an obstacle. The obstacle shown in these figures is not included in the dynamics simulation. It is only added to the animation in order to illustrate the collision successfully avoided by the reactive step sequence planner.

4.3.4 Higher Level Behavior

Some tasks the robot should execute have a longer planning horizon than one step cycle. Examples include stair climbing or walking a certain distance. Such tasks are implemented in the real-time controller as specialized "skills." There always is exactly one active skill and a reference to the next skill to be executed.

The basic structure of the step sequence planning remains unaltered, but two additional functions are called during step sequence planning. The first function is executed before the standard step sequence planner is called. This enables the skill to set or modify walking parameters such as step length or stair height. The second function is called after the standard planner, enabling the skill to modify the results according to special requirements.

All skills are parameterized, allowing the robot to perform a large number of similar actions using one skill. The implemented skills are listed in the following:

Normal Walking This is the default skill active during normal walking. No parameters are modified.

Fixed Distance Walking This skill enables the robot to walk a certain distance, starting and stopping in a standing position. This behavior is parameterized by the desired walking distance, the maximum speed and the desired acceleration.

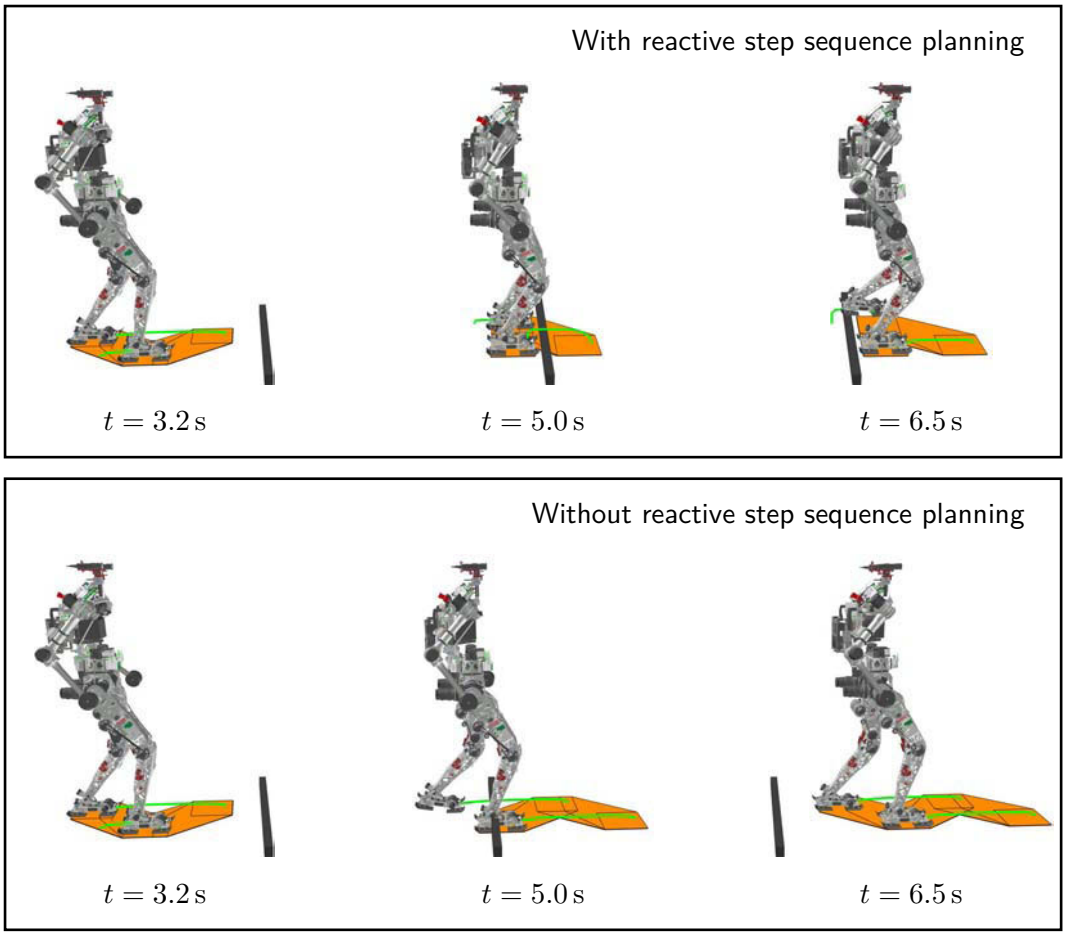


Figure 4.7: The top row shows an example of reactive step sequence planning for stepping over a small obstacle. For comparison, the step sequence generated without an obstacle is shown in the bottom row. Here the obstacle is not present in the dynamics simulation. It is only included in the visualization to show where a collision would have occurred.

Climb on/off Platform Using this skill, the robot can climb up a single step or onto a platform or descend from a platform or step. The motion is parameterized by the position and height of the platform.

4.4 Coordinate Systems and Task-Space Definition

A large number of kinematic and dynamic constraints must be taken into account during the generation of *physically feasible* walking trajectories. Kinematic constraints ensure sufficient ground clearance of the swing leg, specify the correct contact configuration and help to avoid self-collision. The unilateral ground contact limits the set of admissible contact forces, leading to dynamic constraints on feasible movements (cf. Chapter 3).

These constraints are, strictly speaking, independent of the mathematical representation of the robot's configuration. However, generating feasible trajectories can be greatly simplified by using an adequate set of task-space coordinates.

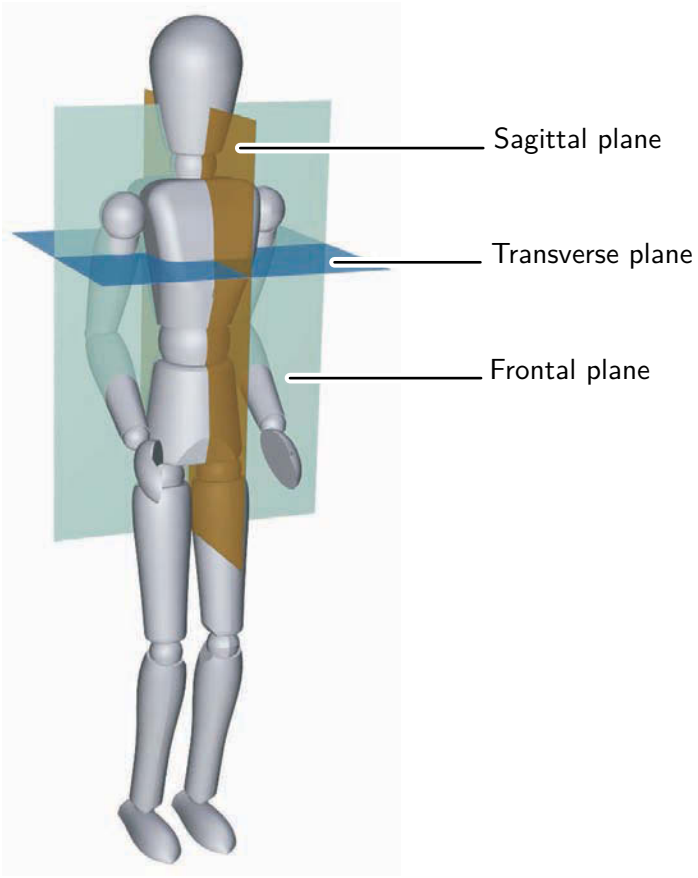


Figure 4.8: Anatomical reference planes of the human body

4.4.1 Coordinate Systems

In the multibody models the position and orientation of coordinate frames are chosen in such a way as to simplify robot simulations. Consequently, the orientation of reference frames in upper body, right and left foot are not aligned when the robot is in an upright standing position, which makes specifying constraints and trajectories complex and non-intuitive.

Therefore, additional reference frames and specialized representations of spatial rotations are introduced. All frames are chosen such that, in a perfectly upright standing position, the unit vector $\boldsymbol{e}_x$ points forward and $\boldsymbol{e}_z$ points upward, i.e., in the opposite direction as gravity.

For each foot, one additional frame is introduced which is fixed to the foot segment for Johnnie and the toe segment for Lola. The origin is located in the center of the sole it is attached to, i.e., in the middle of the entire foot for Johnnie and the middle of the toe segment for Lola. These coordinate systems are denoted by the indices RF for the right and LF for the left foot.

An additional coordinate system with index T is introduced at the origin of the upper body. Figure 4.9 illustrates the location and orientation of foot and upper body frames.

Calculations involving robot dynamics require an inertial reference frame. If only one reference frame were used, calculations would be carried out relative to a

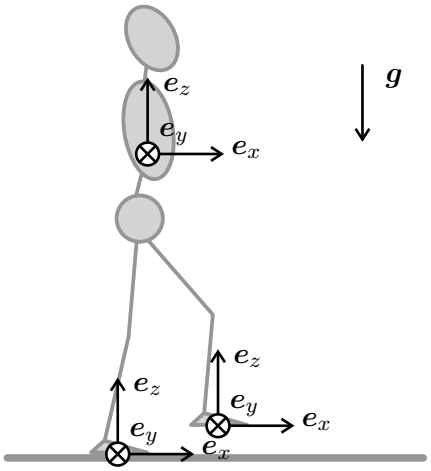


Figure 4.9: Additional coordinate systems for planning and control

more and more distant reference point, potentially leading to large numerical errors. Therefore, a new reference frame I_i is introduced for every step i. When the swing leg touches the ground, the inertial reference frame is placed below the new stance foot, such that it coincides with the planning frame for that foot when there is full surface contact and the foot is not deformed. Figure 4.10 shows a sequence of foot steps and corresponding reference frames. In the following, I signifies the current reference frame I_i, if not explicitly stated otherwise.

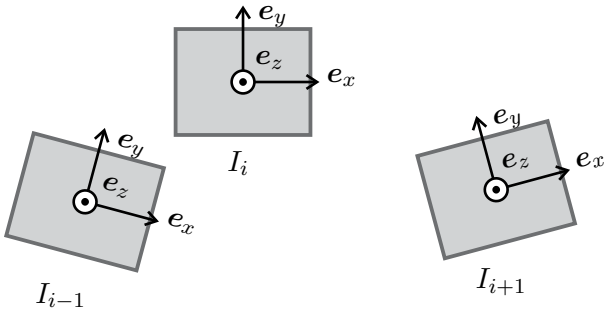


Figure 4.10: One inertial reference frame is introduced for each step.

While the walking control mainly uses the current reference frame I_i, the robot also estimates its position relative to the first frame I_0 by odometry, i.e., by updating the position and rotation of I_i relative to I_0 every time the stance leg switches. This establishes a common frame of reference (I_0) between walking control and computer vision system during autonomous locomotion (cf. Chapter 6).

4.4.2 Task-Space Definition

The task-space coordinates $\boldsymbol{x}(\boldsymbol{q})$ are composed of a sub-vector $\boldsymbol{x}_R(\boldsymbol{q}_J)$ depending only on the joint angles $\boldsymbol{q}_J$ and a sub-vector $\boldsymbol{x}_I(\boldsymbol{q})$ that also depends on the upper body position and orientation. Table 4.2 lists the components of $\boldsymbol{x}$ used for the two robots.

Table 4.2: Components of Johnnie's and Lola's task-space coordinates

Components used for both robots	
${}_T\boldsymbol{r}_{\mathrm{CoG-RF}}$	Right foot TCP relative to CoG in torso planning frame.
${}_T\boldsymbol{r}_{\mathrm{CoG-LF}}$	Left foot TCP relative to CoG in torso planning frame.
${}_T\boldsymbol{s}_{\mathrm{TRF}}$	Orientation of right foot relative to upper body.
${}_T\boldsymbol{s}_{\mathrm{TLF}}$	Orientation of left foot relative to upper body.
${}_I\boldsymbol{r}_{\mathrm{CoG}}$	Robot CoG in inertial planning frame.
$\boldsymbol{\varphi}_{\mathrm{T}}$	Upper Body orientation relative to inertial planning frame.
Components used only for Johnnie	
$q_{J,\mathrm{RA}}$	Right arm joint angle.
$q_{J,\mathrm{LA}}$	Left arm joint angle.
Components used only for Lola	
$\boldsymbol{r}_{\mathrm{CoG,RACoG}}$	Right arm CoG relative to robot CoG in torso planning frame.
$\boldsymbol{r}_{\mathrm{CoG,LACoG}}$	Left arm CoG relative to robot CoG in torso planning frame.
$q_{J,\mathrm{PR}}$	Pelvis rotation.
$q_{J,\mathrm{PA}}$	Pelvis adduction.
$q_{J,\mathrm{RTF}}$	Right toe flexion.
$q_{J,\mathrm{LTF}}$	Left toe flexion.

For both robots, $\boldsymbol{x}_I$ includes the absolute position of the center of gravity and the upper body orientation:

$$\boldsymbol{x}_I = \begin{pmatrix} {}_I\boldsymbol{r}_{\mathrm{CoG}} \\ \boldsymbol{\varphi}_{\mathrm{T}} \end{pmatrix} \tag{4.5}$$

Since center of gravity dynamics dominate overall system dynamics, directly controlling this quantity simplifies both calculating feasible reference motions and maintaining balance.

For Johnnie, the position of the feet relative to the center of gravity and the orientation of the feet relative to the upper body are added, together with the arm joint angles:

$$\boldsymbol{x}_R^T = \begin{pmatrix} {}_T\boldsymbol{r}_{\mathrm{CoG,RF}}^T & {}_T\boldsymbol{r}_{\mathrm{CoG,LF}}^T & {}_T\boldsymbol{s}_{\mathrm{TRF}}^T & {}_T\boldsymbol{s}_{\mathrm{TLF}}^T & q_{J,\mathrm{RA}} & q_{J,\mathrm{LA}} \end{pmatrix} \tag{4.6}$$

Coordinates specifying the foot and center of gravity trajectories are the same for Lola. However, arm trajectories are specified in Cartesian coordinates and the pelvis and toe joint angles are added:

$$\boldsymbol{x}_R^T = \begin{matrix} ({}_T\boldsymbol{r}_{\mathrm{CoG,RF}}^T & {}_T\boldsymbol{r}_{\mathrm{CoG,LF}}^T & {}_T\boldsymbol{s}_{\mathrm{TRF}}^T & {}_T\boldsymbol{s}_{\mathrm{TLF}}^T & \boldsymbol{r}_{\mathrm{CoG,RACoG}}^T & \cdots \\ \cdots & \boldsymbol{r}_{\mathrm{CoG,LACoG}}^T & q_{J,\mathrm{PR}} & q_{J,\mathrm{PA}} & q_{J,\mathrm{RTF}} & q_{J,\mathrm{LTF}}) \end{matrix} \tag{4.7}$$

By controlling the foot position and orientation, premature ground contact of the swing leg and collisions between swing and stance foot are easily avoided.

For both robots, there is a unique mapping from $\boldsymbol{x}$ to $\boldsymbol{q}$. For Lola, however, not all coordinates are actively tracked. For example, by deactivating tracking for the

pelvis joints, these can be used for minimizing joint speeds and avoiding joint limits (cf. Section 5.4). It is nevertheless useful to have a unique specification of the ideal kinematic configuration, e.g., for initializing the robot controller and as a reference for null-space optimization.

4.4.3 Relative Foot Orientation

In the field of robotics, orientations are often parameterized using Euler or Cardan angles, the axis-angle representation or unit quaternions. However, these representations are not well suited for specifying feasible foot trajectories. For example, forward foot roll during curve walking would lead to a complex and non-intuitive pattern of Euler angles.

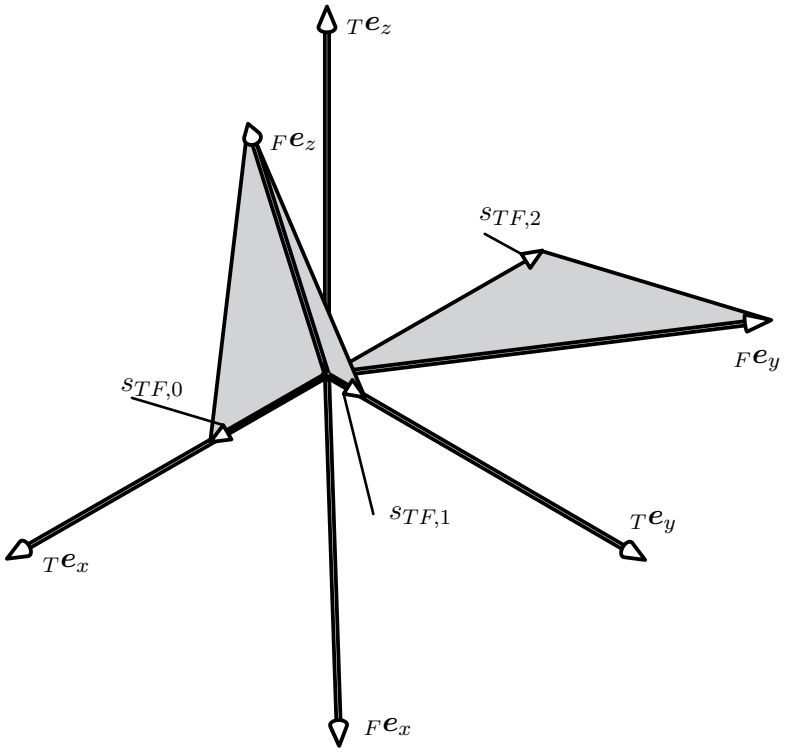


Figure 4.11: Coordinates specifying relative orientation of foot and upper body frames

Instead, a representation is chosen that enables direct specification of the angles corresponding to forward roll, curve walking and walking sideways. This representation was developed by Löffler for the robot Johnnie and is based on direction cosines between unit vectors of two reference frames. Figure 4.11 illustrates the geometric relationships. With F signifying either the right (RF) or left foot (LF), the orientation is parameterized by:

$$s_{TF,0} = \boldsymbol{e}_z^T \boldsymbol{A}_{FT} \boldsymbol{e}_x \tag{4.8}$$

$$s_{TF,1} = \boldsymbol{e}_z^T \boldsymbol{A}_{FT} \boldsymbol{e}_y \tag{4.9}$$

$$s_{TF,2} = \boldsymbol{e}_y^T \boldsymbol{A}_{FT} \boldsymbol{e}_x \tag{4.10}$$

The term $s_{TF,0}$ corresponds to an up/down movement of the toe, $s_{TF,1}$ specifies a sideways tilt and $s_{TF,2}$ the rotation required for curve walking.

4.4.4 Absolute Upper Body Orientation

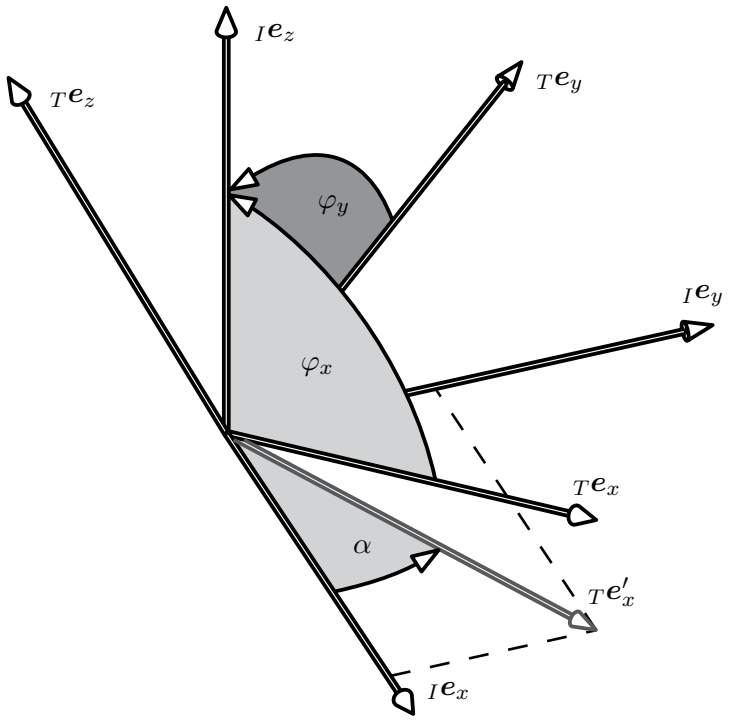


Figure 4.12: Angles specifying the upper body orientation

Löffler proposed a special representation of the upper body rotation relative to an inertial reference frame [60]. Using this representation, the upper body's inclination in the sagittal and the coronal plane, as well as the rotation about the gravity vector can be specified independently. The kinematics of this representation will be explained with reference to Figure 4.12. The axes of the inertial reference frame are denoted by ${}_I\boldsymbol{e}_x\,, {}_I\boldsymbol{e}_y\,, {}_I\boldsymbol{e}_z$ and those of the torso frame by ${}_T\boldsymbol{e}_x\,, {}_T\boldsymbol{e}_y\,, {}_T\boldsymbol{e}_z$. The three angles $\boldsymbol{\varphi}_T = (\varphi_x\ \varphi_y\ \alpha)^T$ are defined as:

$$\varphi_x = \angle({}_I\boldsymbol{e}_z\,, {}_T\boldsymbol{e}_x) \tag{4.11}$$
$$\varphi_y = \angle({}_I\boldsymbol{e}_z\,, {}_T\boldsymbol{e}_y) \tag{4.12}$$
$$\alpha = \angle({}_I\boldsymbol{e}_x\,, {}_T\boldsymbol{e}'_x) \tag{4.13}$$

The vector ${}_T\boldsymbol{e}'_x$ is the projection of ${}_T\boldsymbol{e}_x$ onto the x-y plane of the inertial reference frame.

The multibody model uses Euler angles, so we must be able to transform $(\varphi_x\ \varphi_y\ \alpha)^T$ into $(\psi\ \vartheta\ \varphi)^T$ and vice-versa. Conversion between the two representations is performed via the rotation matrix $\boldsymbol{A}_{IT}$. For the sake of simplicity, the upper body inclination is assumed to be less than 90°, which seems reasonable for biped walking.

Calculating φ_x, φ_y from the rotation matrix is straightforward:

$$\varphi_x = \operatorname{acos}({}_I\boldsymbol{e}_z^T \, {}_T\boldsymbol{e}_x) = \operatorname{acos}(\boldsymbol{e}_z^T \boldsymbol{A}_{IT} \boldsymbol{e}_x) \tag{4.14}$$

$$\varphi_y = \operatorname{acos}({}_I\boldsymbol{e}_z^T \, {}_T\boldsymbol{e}_y) = \operatorname{acos}(\boldsymbol{e}_z^T \boldsymbol{A}_{IT} \boldsymbol{e}_y) \tag{4.15}$$

Using

$${}_T\boldsymbol{e}'_x = \begin{pmatrix} \boldsymbol{e}_x^T \boldsymbol{A}_{IT} \boldsymbol{e}_x & \boldsymbol{e}_y^T \boldsymbol{A}_{IT} \boldsymbol{e}_x & 0 \end{pmatrix}^T, \tag{4.16}$$

the third angle α is calculated as

$$\alpha = \operatorname{atan2}\left(\boldsymbol{e}_y^T \boldsymbol{A}_{IT} \boldsymbol{e}_x, \boldsymbol{e}_x^T \boldsymbol{A}_{IT} \boldsymbol{e}_x\right) \tag{4.17}$$

The function $\operatorname{atan2}(y, x)$ is a variant of $\operatorname{atan}(y/x)$ that returns the correct angle between the positive x-axis and the point (x, y) for all values of x, y other than $x = y = 0$.

Löffler proposed calculating the rotational matrix from $\boldsymbol{\varphi}_T$ by solving a set of quadratic equations. The author instead proposes a closed form solution for calculating the rotational matrix $\boldsymbol{A}_{TI} = ({}_I\boldsymbol{e}_x \;\; {}_I\boldsymbol{e}_y \;\; {}_I\boldsymbol{e}_z)$ directly. For $\alpha = 0$ the result is immediately obtained as:

$${}_I\boldsymbol{e}_x|_{\alpha=0} = \begin{pmatrix} \mathrm{s}\,\varphi_x \\ 0 \\ \mathrm{c}\,\varphi_x \end{pmatrix} \tag{4.18}$$

$${}_I\boldsymbol{e}_y|_{\alpha=0} = \begin{pmatrix} 0 \\ \mathrm{s}\,\varphi_y \\ \mathrm{c}\,\varphi_y \end{pmatrix} \tag{4.19}$$

$${}_I\boldsymbol{e}_z|_{\alpha=0} = {}_I\boldsymbol{e}_x \times {}_I\boldsymbol{e}_y = \begin{pmatrix} -\mathrm{c}\,\varphi_x \,\mathrm{s}\,\varphi_y \\ -\mathrm{s}\,\varphi_x \,\mathrm{c}\,\varphi_y \\ \mathrm{s}\,\varphi_x \,\mathrm{s}\,\varphi_y \end{pmatrix} \tag{4.20}$$

Here $\mathrm{c}\,x$ and $\mathrm{s}\,x$ are used to abbreviate $\cos x$ and $\sin x$. Since α is a rotation about the z-axis of the coordinate system I, the final result is obtained by chaining $\boldsymbol{A}_{TI}|_{\alpha=0}$ with an elementary rotation about the z-axis:

$$\boldsymbol{A}_{TI} = \boldsymbol{A}_z(\alpha) \; \boldsymbol{A}_{TI}|_{\alpha=0} \tag{4.21}$$

The resulting matrix is listed in Appendix D, Equation (D.5).

4.5 Foot Trajectory Generation

Foot trajectories are defined by the relative position of the foot reference point from the current planning frame and the rotation of the feet relative to the upper body. The basic concept for designing foot trajectories follows the approach proposed by Löffler for Johnnie [60]. However, arbitrary walking directions are considered in this thesis, while Johnnie's original controller did not support walking sideways. Also,

Lola's controller supports the use of active toe joints. Additionally, the extensions listed in Section 4.9 were added to improve walking performance.

Trajectory Representation

Foot trajectories are parameterized by piecewise quintic polynomials. For a component p with n phases, the trajectory is given by:

$$p(t) = \begin{cases} \sum_{j=0}^{5} a_{0,j}(t-t_0)^j & \text{for} \quad t_0 \leq t < t_1 \\ \sum_{j=0}^{5} a_{1,j}(t-t_1)^j & \text{for} \quad t_1 \leq t < t_2 \\ \dots \\ \sum_{j=0}^{5} a_{n-1,j}(t-t_{n-1})^j & \text{for} \quad t_{n-1} \leq t < t_n \end{cases} \tag{4.22}$$

The coefficients $a_{i,j}$ are calculated from positions, velocities and accelerations at phase boundaries, which guarantees $\mathcal{C}^2$-smooth trajectories:

$$p(t_i) = a_{0,i} \tag{4.23}$$
$$\dot{p}(t_i) = a_{1,i} \tag{4.24}$$
$$\ddot{p}(t_i) = 2a_{2,i} \tag{4.25}$$
$$p(t_{i+1}) = \sum_{j=0}^{5} a_{i,j}(t-t_i)^j \tag{4.26}$$
$$\dot{p}(t_{i+1}) = \sum_{j=0}^{4} (j+1)a_{i,j+1}(t-t_i)^j \tag{4.27}$$
$$\ddot{p}(t_{i+1}) = \sum_{j=0}^{3} \frac{(j+2)!}{j!} a_{i,j+2}(t-t_i)^j \tag{4.28}$$

Equations (4.23) to (4.25) directly yield $a_{0,i}, a_{1,i}$ and $a_{2,i}$. The remaining parameters are obtained by numerically solving (4.26) to (4.28). Overall, calculation of these coefficients has a negligible computational cost.

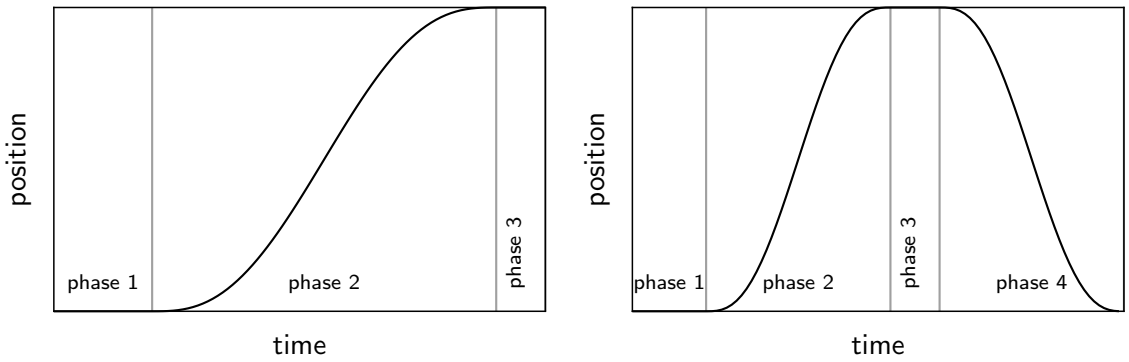
Translational Trajectory

The horizontal component of the swing foot trajectory uses three phases per step. The first phase extends until $\Delta T_{\text{swing,lag}}$ after the double support phase ends and the third phase starts $\Delta T_{\text{swing,lead}}$ before the next step begins. The foot is moved to the next foothold during the second phase and does not move in the horizontal direction during the first and third phases. Figure 4.13(a) shows a schematic plot of one component of a horizontal swing foot trajectory.

Vertical Trajectory

Vertical motion of the swing foot is divided into four phases. During double support the foot is at ground level, during the third phase it remains at a constant height above ground and the remaining phases are used for lifting and lowering the foot. A schematic plot of the resulting vertical foot trajectory is shown in Figure 4.13(b).

The parameters $\Delta T_{\text{swing,lead}}$ and $\Delta T_{\text{swing,lag}}$ determine if lift-off and/or landing are vertical and how soon the desired ground clearance z_{swing} is reached.



(a) Horizontal trajectory (b) Vertical trajectory

Figure 4.13: Design of the basic horizontal (a) and vertical (b) swing foot trajectories

Rotational Trajectory

Foot rotation about the vertical axis is added for curve walking. The trajectory is designed with the same approach used for horizontal translation of the foot. Additionally, the basic horizontal and vertical foot trajectories described above are modified when forefoot roll and heel strike are used in order to satisfy the rolling condition at the toe or heel and maintain the original velocity and position of the foot relative to the ground.

Although not strictly necessary for walking, foot rotation orthogonal to the walking direction has several advantages. During lift-off, foot rotation increases the maximum possible step length by reducing ankle flexion, which is a limiting factor especially for Johnnie. The same mechanism is used for Lola, where active toe joints additionally allow the robot to maintain area contact during heel-off.

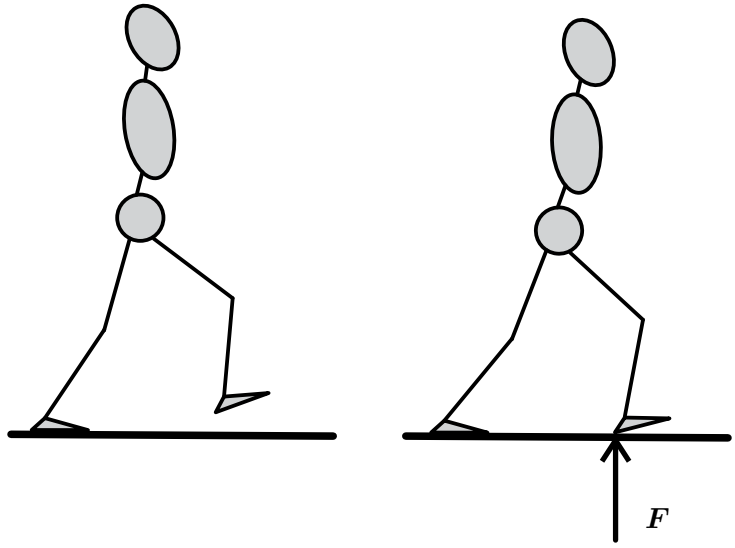
During the swing phase, foot rotation ensures initial contact with the heel. Both Löffler's walking controller [60] and Honda's robots[1] also use swing foot rotation. Nevertheless, an analysis of its influence on walking stability has, to the author's knowledge, not been published previously.

Reference trajectories are planned so that, for longer steps, the center of pressure is at the robot's heel during initial contact. If the heel is lower than the toe, the robot will contact the ground with the heel first even in the presence of an upper body inclination error. If, however, the foot remains horizontal to the ground on its ideal trajectory, initial contact will occur either with the heel or the toe, depending on the sign of the upper body inclination error. That is, infinitely small upper body inclination errors lead to finite errors in the position of the resulting contact force. This is illustrated in Figure 4.14.

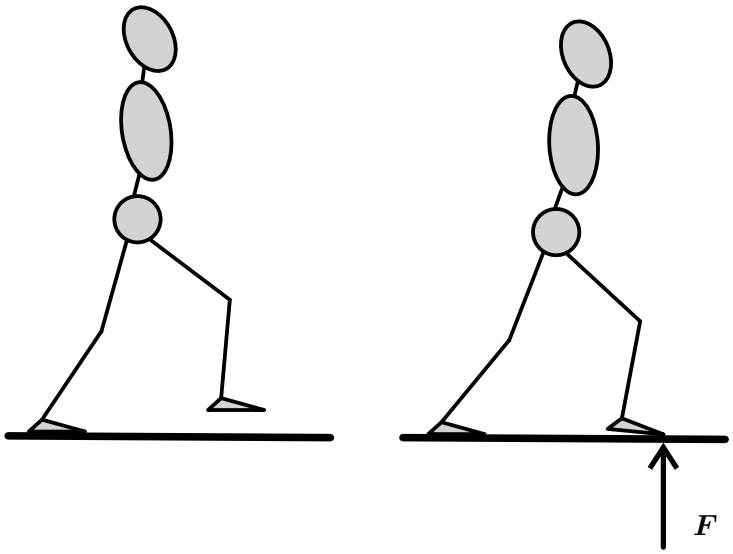
Toe Trajectory

For Lola, a heuristic reference trajectory for the toe joint angle is calculated using quintic polynomials. The trajectory is set to reach the maximum toe joint angle in the middle of the initial swing phase. The peak toe joint angle is a linear function of

1 This is from personal communication with a senior engineer in Honda's robotics department.



(a) Foot rotation ensures heel-strike



(b) A flat-footed gait can lead to initial contact with the toe

Figure 4.14: A gait with heel-strike (top) is more robust against upper body inclination errors than a flat-footed gait (bottom), because the location of the force at initial contact does not change.

the step length:

$$\varphi_{\text{toe,peak}} = \begin{cases} 0 & \text{for} \quad L_{\text{step}} < L_{\text{foot}} \\ \frac{\varphi_{\text{toe,max}}}{L_{\text{max}} - L_{\text{foot}}}(L_{\text{step}} - L_{\text{foot}}) & \text{for} \quad L_{\text{step}} \geq L_{\text{foot}} \end{cases} \tag{4.29}$$

Here L_{foot} is the foot length, L_{max} is the maximum step length and $\varphi_{\text{toe,max}}$ the largest toe angle to be used. For the results shown in this thesis, $\varphi_{\text{toe,max}} = 40°$ to $60°$ and $L_{\text{max}} = 0.6$ to $0.8\,\text{m}$.

Heuristically determining the optimal speed and acceleration at the peak toe joint angle is difficult and error prone. Therefore, only the joint angle is set and free

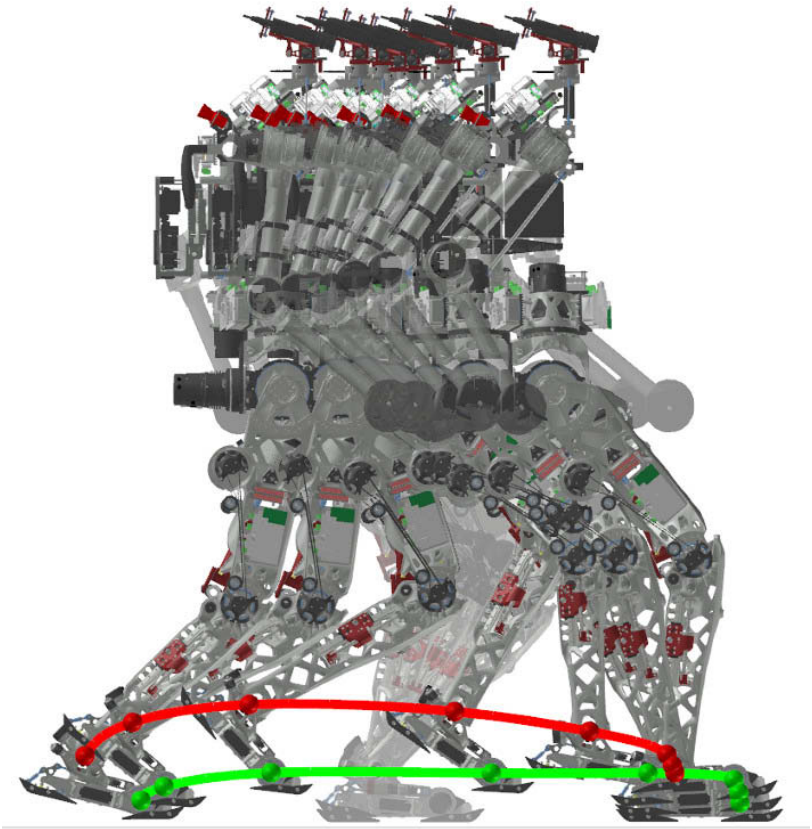


Figure 4.15: Example of a swing foot trajectory for Lola. The illustration shows snapshots of the robot every 0.16 s. The reference leg is drawn solid, other bodies transparent. The trajectory of the heel frame is shown in red, that of the toe in green.

parameters are determined by minimizing the required acceleration. To simplify the optimization problem, the position p_i, velocity $\dot{p}_i$ and acceleration $\ddot{p}_i$ at phase boundaries t_i are chosen as independent variables. The optimization problem can then be written as:

$$\begin{aligned} \phi(\boldsymbol{x}_p) &= \frac{1}{2}\int \ddot{p}^2 dt \rightarrow \min! \\ \boldsymbol{G}\boldsymbol{x}_p &= \boldsymbol{g}_0 \end{aligned} \tag{4.30}$$

Here $\boldsymbol{x}_p$ is the vector of all boundary values $p_i, \dot{p}_i, \ddot{p}_i$. Due to the choice of variables, $\boldsymbol{G}$ is a binary matrix and the problem can be solved efficiently in the subspace of free parameters. The same method of computing trajectories with minimum acceleration can also be used for other components.

An example of a resulting swing foot trajectory for Lola is shown in Figure 4.15.

4.6 Center of Gravity Trajectory Generation

This section presents a new method for real-time walking pattern generation that calculates stable walking patterns from given footstep locations. The system consists

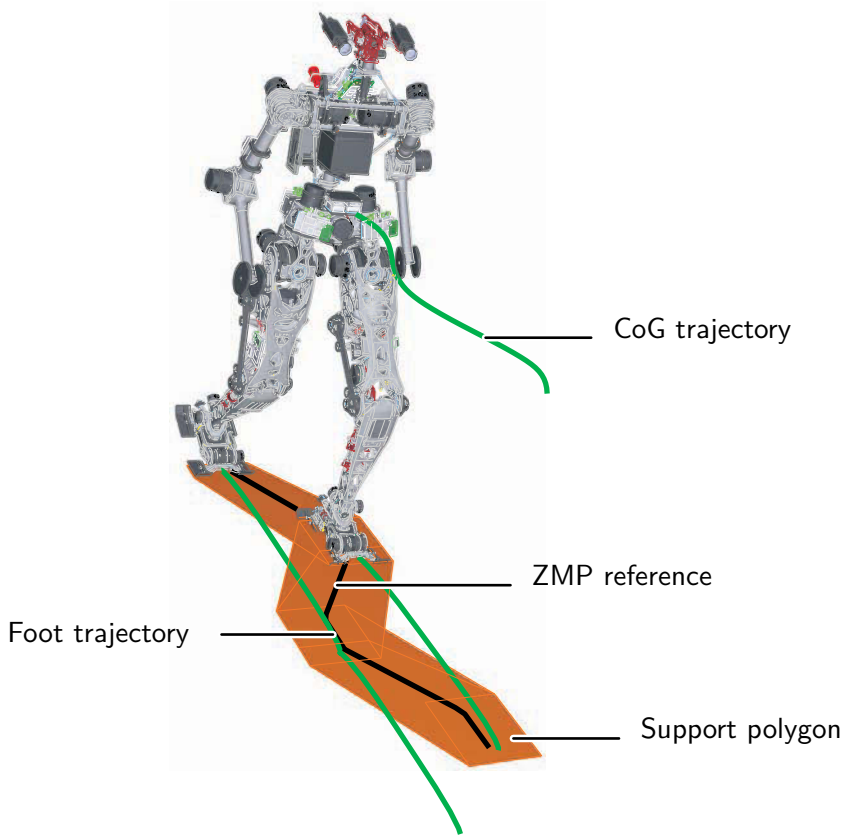


Figure 4.16: Visualization of real-time trajectory planning. Foot and CoG trajectories shown in green, ZMP reference trajectory in black and support polygons in orange.

of constraint calculation, contact force trajectory generation and a method for solving boundary value problems (BVP) for a sequence of walking steps. The author previously published the basic concept of this method in [6]. This section describes an extended version that can generate better trajectories for curve walking and plans contact force trajectories in 2-D, taking the couplings between sagittal and lateral plane into account.

4.6.1 Related Work

Optimization-based methods using comprehensive multibody models have been presented in [3, 9, 18]. Due to the high computational cost, these methods can only be used offline. To facilitate online planning, a large database of possible gaits must be generated. If the solution space for a larger number of walking parameters is to be sampled, this approach suffers from the "curse of dimensionality." In fact, the author implemented a system for generating walking trajectories using parameter optimization [9]. This approach was very useful for generating realistic joint loads in an early design stage of Lola and for systematically exploring different feasible trajectories. Especially for autonomous walking, however, the approach proved impractical, since providing enough step primitives leads to a combinatorial explosion

of the gait database. For Lola, typical ranges of the basic walking parameters are:

$$\begin{aligned} L_x &= -20 \ldots 66\,\text{cm} \\ L_y &= -20 \ldots 20\,\text{cm} \\ \varphi_{\text{step}} &= -20 \ldots 20\,\text{deg} \\ T_{\text{step}} &= 0.7 \ldots 1.2\,\text{s} \end{aligned}$$

A reasonably dense sampling of this parameter space should include gaits for step lengths every 2 cm, turning angles every 2 deg and step periods every 0.1 s. This leads to a total of 8.6×10^4 different walking patterns. This can be reduced by, e.g., approximating the trajectory for 20 cm steps using an interpolation between 18 cm and 22 cm steps. However, an even larger number of transitional gaits for switching from one set of parameters to the next must be calculated and stored. It is not possible to generate these transitional walking patterns by blending from one set of parameters to the next, since the resulting trajectory often is not physically feasible.

Most authors therefore propose using real-time planning to be able to respond more flexibly to changing commands and environments. Most work is based on a simplified "inverted pendulum" model [41, 45, 61, 120, 121]. Proposed algorithms using this model include analytical solutions [69], numerical solutions by finite difference approximation [41], a kind of shooting method [120, 121] and a method based on model predictive control [45]. A more detailed review of related work can be found in [121].

The author proposes a method using the same basic approach as other real-time walking pattern generators mentioned above. The novelty of the method is the algorithm for generating reference force patterns and the approach towards solving the equations of motion in real-time, which can handle more complex equations of motion than previous methods. More specifically, the proposed method can solve the trajectory generation problem for a non-constant height of the center of gravity and arbitrary right hand sides.

4.6.2 Problem Statement and Analysis

There are two fundamental difficulties in walking pattern generation. First, feasible contact forces and moments are strongly limited, leading to unilateral constraints (cf. Chapter 3). Since dealing with inequalities in motion planning is computationally expensive, the usual approach is to first generate feasible force and moment reference patterns, effectively eliminating the inequalities. The remaining task is then to calculate robot trajectories, often only the CoG trajectory, corresponding to the given reference forces and moments. In this approach, forces are inputs and the output is the robot's CoG trajectory. However, the corresponding initial value problem (IVP) is numerically unstable, which is the second difficulty. The obvious solution is to solve a BVP instead of an IVP, thus preventing divergence.

In real-time planning, it is necessary to connect a newly calculated trajectory to the one currently being executed. To avoid excessive forces that could damage and/or destabilize the robot, connections should be $\mathcal{C}^2$-smooth, i.e., initial position, velocity and acceleration must equal those of the trajectory currently being executed.

Additionally, constraints on the final CoG position and/or velocity are included to achieve convergence towards a desired orbit or state. This is an ill-posed problem, since more than two conditions are specified for a second order ODE. Therefore, to enable $\mathcal{C}^2$-smooth trajectories, the reference forces and moments must be modified to introduce additional degrees of freedom required to obtain a well-posed problem.

Kajita [45] proposed an interesting solution to this problem. By addressing walking pattern generation as a tracking control problem for a given ZMP trajectory, smooth solutions are obtained. However, this "preview control" method does not allow ZMP trajectory modifications within the preview period, which is typically set to about 1.6 s. This delay is a significant drawback for vision-guided walking. Furthermore, in the author's experience, large changes in the desired step length can lead to very large deviations from the ZMP reference.

In an initial implementation of the method proposed here, a statically stable CoG position was used as boundary condition at the end of the reference trajectory [6]. While this approach worked well, convergence towards a periodic gait could be slow. Furthermore, this boundary condition is not well suited for curved paths, since the curvature is not taken into account.

The author therefore proposes to first plan a periodic CoG trajectory for the third and fourth step. For this portion of the trajectory the boundary conditions are:

$$\begin{aligned} \boldsymbol{r}(t_{\text{step},4}) - \boldsymbol{r}(t_{\text{step},2}) &= \boldsymbol{\Delta r}_{\text{step},2} + \boldsymbol{\Delta r}_{\text{step},3} \\ \dot{\boldsymbol{r}}(t_{\text{step},4}) - \dot{\boldsymbol{r}}(t_{\text{step},2}) &= \boldsymbol{0} \end{aligned} \tag{4.31}$$

Here index 0 is assigned to the first step of the new trajectory and $t_{\text{step},k}$ is the instant when the k-th step begins. Note that this problem is well defined and no modification of contact force trajectories is necessary.

The initial state of the periodic solution is used as a boundary condition for the first two steps. Denoting the trajectory currently being executed as $\boldsymbol{r}_{\text{cur}}$ and the periodic trajectory as $\boldsymbol{r}_{\text{per}}$, the boundary conditions are given by:

$$\begin{aligned} \boldsymbol{r}(t_{\text{step},0}) &= \boldsymbol{r}_{\text{cur}}(t_{\text{step},0}) \\ \boldsymbol{r}(t_{\text{step},2}) &= \boldsymbol{r}_{\text{per}}(t_{\text{step},2}) \\ \dot{\boldsymbol{r}}(t_{\text{step},0}) &= \dot{\boldsymbol{r}}_{\text{cur}}(t_{\text{step},0}) \\ \dot{\boldsymbol{r}}(t_{\text{step},2}) &= \dot{\boldsymbol{r}}_{\text{per}}(t_{\text{step},2}) \end{aligned} \tag{4.32}$$

Periodic reference trajectories are also used for Toyota's and Honda's robots [116, 121]. For Honda's robot, Takenaka proposed using not position and velocity, but only the "divergent component" of the trajectory in the boundary condition for the endpoint. This is a very interesting approach, since it avoids over-constraining the system, which can lead to excessive changes in the reference ZMP in order to satisfy the boundary conditions. This is especially true when a trajectory for only one step is generated, as proposed by Takenaka. Contrary to this approach, we propose generating a trajectory for two instead of only one step, spreading changes of the ZMP over a longer period. In the author's experience, the trajectory for the first step, which is the only part ever executed, is very similar for both types of boundary conditions, if the planning horizon is longer than one step.

4.6.3 Center of Gravity Dynamics

Calculating feasible trajectories taking a comprehensive multibody model into account is possible [3, 9, 18], but computationally expensive. Currently, real-time calculation is only feasible using simplified models. The required modeling depth is reduced if details, such as individual joint torques, are not considered during trajectory generation. To determine feasible trajectories, it is sufficient to study the overall linear and angular momentum of the robot (cf. Chapter 3):

$$m\ddot{\boldsymbol{r}}_{\mathrm{CoG}} = \boldsymbol{F} + m\boldsymbol{g} \tag{4.33}$$

$$\dot{\boldsymbol{L}}^{\mathrm{CoG}} = \boldsymbol{T} - \boldsymbol{r}_{\mathrm{CoG}} \times \boldsymbol{F} \tag{4.34}$$

Here m is the robot gravity, $\boldsymbol{g}$ the gravity vector, $\boldsymbol{L}^{\mathrm{CoG}}$ the angular momentum about the CoG and $\boldsymbol{F}$, $\boldsymbol{T}$ the total contact force and moment acting on the robot. Eqs. (4.33)-(4.34) are exact but (4.34) is strongly nonlinear (in $\boldsymbol{q}$), making solutions numerically expensive. By assuming $\dot{\boldsymbol{L}}^{\mathrm{CoG}} \approx \boldsymbol{0}$ the much simpler EoM for the CoG-dynamics can be derived:

$$m\ddot{z} - mg = F_z \tag{4.35}$$

$$mz\ddot{x} - mx(\ddot{z} + g) = T_y \tag{4.36}$$

$$mz\ddot{y} - my(\ddot{z} + g) = -T_x \tag{4.37}$$

Since (4.36) and (4.37) are decoupled if $z(t)$ is assumed to be known, the further discussion will be limited to the lateral direction. In fact, (4.36) and (4.37) can be derived and interpreted in a number of ways. See [24, 65] for an interesting discussion of the planar case.

Eq. (4.37) can be interpreted as the EoM of a single point mass coinciding with the robot's CoG and is surprisingly accurate for slow and moderate walking speeds. However, at higher walking speeds, the influence of swing leg motion increases, especially in the sagittal direction. This is especially true for the robots Johnnie and Lola, since the legs account for approximately 60% of the total weight [25, 62].

To take this into account, simplified models with additional point masses representing the legs have been proposed. Takenaka [120] proposes a model with one additional point mass per leg while Park [85] proposes only one additional point mass representing the swing leg. The modeling error of the single point mass model is mainly due to swing leg motion, suggesting a two-mass model. However, since stance and swing leg switch for planning horizons of more than one step, we adopt a model with one additional point mass per leg. The three point mass system is described by:

$$\begin{aligned} m_b z_b \ddot{y}_b - m_b y_b(\ddot{z}_b + g) = -T_x &+ m_l y_{l1}(\ddot{z}_{l1} + g) - m_l z_{l1}\ddot{y}_{l1} \\ &+ m_l y_{l2}(\ddot{z}_{l2} + g) - m_l z_{l2}\ddot{y}_{l2} \end{aligned} \tag{4.38}$$

where index b denotes the body mass point, indices $l1, l2$ refer to the first and second leg mass points and m_l is the mass of one leg mass point.

Since the swing leg trajectory is calculated independently, (4.38) can be written in

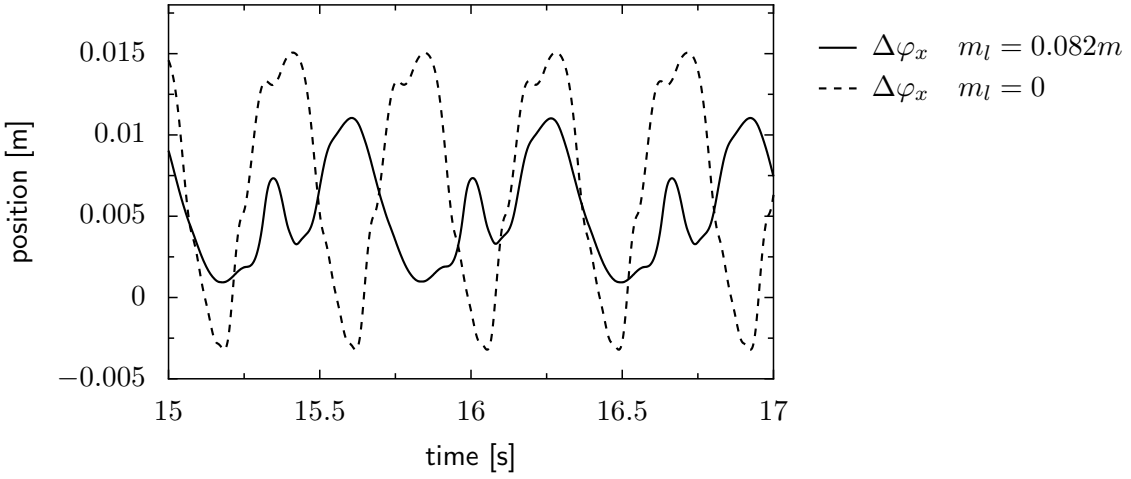


Figure 4.17: A three point mass model leads to decreased upper body inclination errors $\Delta\varphi_x$ when compared to a simple lumped mass model ($m_l = 0$). The plot shows results from a dynamics simulation of Johnnie.

the form of (4.37) by defining the modified right hand side:

$$\tilde{T}_x := T_x - m_l y_{l1}(\ddot{z}_{l1} + g) + m_l z_{l1}\ddot{y}_{l1} - m_l y_{l2}(\ddot{z}_{l2} + g) + m_l z_{l2}\ddot{y}_{l2} \tag{4.39}$$

Figure 4.17 shows simulation results for Johnnie's upper body inclination for $m_l = 0$ and $m_l = 8.2\%$ at a walking speed of 2.5 km/h. Evidently, oscillations of the upper body due to swing leg motion are significantly reduced by using the improved model.

4.6.4 ZMP Reference Trajectory

This section presents a method for automatically generating ZMP reference trajectories from the foot geometry, footstep locations and a list of active contacts for each gait phase.

Because of the unilateral ground contact, the ZMP $\boldsymbol{r}_p$ must remain within the convex hull of the support polygon at all times (cf. Chapter 3):

$$\boldsymbol{r}_p(t) \in \mathrm{co}\left(\{\boldsymbol{P}_i(t) | i \in I_c(t)\}\right) \tag{4.40}$$

Here $\boldsymbol{P}_i(t)$ denotes the position of the i-th contact and $I_c(t)$ is the set of active contacts at time t. The set of active contacts is determined during step sequence planning. Since $I_c(t)$ is constant for each phase of gait, the support polygon is a piecewise constant function of time.

Theoretically, any feasible ZMP trajectory should be acceptable. In reality, however, contact force and corresponding center of gravity trajectories have a strong influence on walking stability and maximum walking speed. The author therefore proposes using trajectories minimizing

$$\begin{aligned} &\frac{1}{2}\int \dot{\boldsymbol{r}}_p^2 dt \rightarrow \min! \\ &\quad \boldsymbol{r}_p(t) \in \mathrm{co}\left(\{\boldsymbol{P}_i(t) | i \in I_c(t)\}\right) \end{aligned} \tag{4.41}$$

Such trajectories have two favorable characteristics: 1) Slowly changing contact forces are easier to track for the contact force control and 2) they lead to smaller center of gravity velocities and smoother walking.

In order to have $\mathcal{C}^2$-smooth CoG trajectories, $\boldsymbol{r}_p$ must be at least $\mathcal{C}^0$, i.e., continuous. Then, the solution to (4.41) is a piecewise linear function. For portions that lie on the support polygon boundary, this is self-evident. For the remaining pieces, this is easily verified by studying the unconstrained problem using calculus of variations.

Since kinks can only occur at boundaries between gait phases, $\boldsymbol{r}_p(t)$ is parameterized as a piecewise linear function with control points $\boldsymbol{p}_k$ at phase boundaries. Using this representation, (4.41) is transformed into a quadratic programming problem:

$$\begin{aligned} \frac{1}{2}\boldsymbol{p}_x^T\boldsymbol{Q}\boldsymbol{p}_x + \frac{1}{2}\boldsymbol{p}_y^T\boldsymbol{Q}\boldsymbol{p}_y \rightarrow \min! \\ \boldsymbol{p}_k \in \operatorname{co}\left(\{\boldsymbol{P}_i(t_k) | i \in I_c(t_k)\}\right) \end{aligned} \tag{4.42}$$

Here $\boldsymbol{p}_x$, $\boldsymbol{p}_y$ are vectors of x and y components of the n control points and $\boldsymbol{Q}$ is a tridiagonal matrix depending only on the control point distribution in time:

$$\boldsymbol{Q} = \begin{pmatrix} -\frac{1}{h_0} & \frac{1}{h_0} & 0 & \dots & & & \\ -\frac{1}{h_0} & \frac{1}{h_0}+\frac{1}{h_1} & -\frac{1}{h_1} & 0 & \dots & & \\ 0 & -\frac{1}{h_1} & \frac{1}{h_1}+\frac{1}{h_2} & -\frac{1}{h_2} & 0 & \dots & \\ \vdots & & \ddots & \ddots & \ddots & & \\ & & & & \dots & 0 & -\frac{1}{h_{n-2}} & \frac{1}{h_{n-2}} \end{pmatrix} \tag{4.43}$$

Here $h_i = t_{i+1} - t_i$ is the length of the i-th phase of gait. For given endpoint positions $\boldsymbol{p}_0$, $\boldsymbol{p}_{n-1}$, (4.42) can be solved using a simple iterative scheme. The optimality conditions for the unconstrained case, $\boldsymbol{Q}\boldsymbol{p}_x = \boldsymbol{Q}\boldsymbol{p}_y = \boldsymbol{0}$, are solved using a modified Gauss-Seidel method. After each iteration, modified control points $\boldsymbol{p}_k$ are projected onto the boundary of the support polygon if they are outside. The strength of this method is that it can handle arbitrary foot geometries. However, convergence can be slow in some situations, even if it is faster than for a general quadratic programming (QP) solver. Therefore, a faster method was also implemented that directly constructs a solution for rectangular feet based on geometric reasoning.

Figure 4.16 shows results of the inequality and ZMP reference calculations for three steps. Inequality constraints are shown in the form of support polygons in orange, the ZMP reference trajectory in black and foot and CoG trajectories in green.

4.6.5 Solving the Boundary Value Problem

In this section we propose two methods for solving the boundary value problem. Only the initial part of the trajectory with conditions (4.32) is presented. Calculating the periodic reference trajectory is similar but does not require modifying ZMP trajectories.

We first present an analytical solution. This method is very fast, but limited to a constant CoG height and a simple inverted pendulum model. The second method calculates an approximate solution using a collocation method. While this approach

is numerically more expensive, it can solve time-varying linear ODEs that arise when the z-component is not constant and can handle arbitrary right hand sides.

Analytical Solution

Most of the time walking is a periodic motion, so (4.37) is a linear ODE where the coefficients are periodically time-varying, i.e., (4.37) corresponds to the HILL equation. For some special choices of $z(t)$, solutions exist, e.g., in the form of MATHIEU functions. In general, however, there is no closed form solution.

It is therefore common to assume $z = \text{const.}$ to allow an analytical solution. For right hand sides composed of basic functions the solution is straightforward. An analytical solution for piecewise polynomial right hand sides is given in [69]. In [6] the author proposed a slightly modified method, using reference moments calculated as shown in Section 4.6.4. For the analytical method, modified boundary conditions are used. The initial position and velocity, along with the final position, are considered, while the final velocity is not constrained.

To enable $\mathcal{C}^2$-smoothly connected trajectories, a control point $T_{x,\text{mod}}$ of the reference moment located at the middle of the first double support phase is modified. Since stability margins are high at this point in the walking cycle, this leads to feasible walking patterns even for large changes in walking parameters, e.g., when the robot is accelerating.

For an interval $t \in [t_i, t_{i+1}]$ of the reference moment the solution of (4.37) with $\ddot{z} = 0, z = \text{const.}$ is given by

$$y_i = a_i e^{\sqrt{\alpha}t} + b_i e^{-\sqrt{\alpha}t} - \frac{T_{x,i}}{mg} - \frac{T_{x,i+1} - T_{x,i}}{mg(t_{i+1} - t_i)}\tau_i \tag{4.44}$$

where $i \in [0 \dots n-1]$, $\alpha := g/z$ and $\tau_i := (t - t_i)$. To obtain a $\mathcal{C}^2$-smooth solution of the BVP for a reference moment with n control points, we demand continuity of y and $\dot{y}$ at the $(n-2)$ interior points $i \in [1 \dots n-2]$. Adding the three boundary conditions for $y|_{t=t_0}, \dot{y}|_{t=t_0}$ and $y|_{t=t_{n-1}}$ leads to a total of $(2n-1)$ linear equations for the $2(n-1)+1$ variables, a_i, b_i and $T_{x,\text{mod}}$, giving a unique solution. Figure 4.18 shows a solution for the sagittal direction for Johnnie accelerating from a previous step length of 20 cm to 55 cm steps. Evidently, the method produces feasible and smooth trajectories even for large and sudden changes in step length.

Solution by Spline Collocation

For general $z_b(t)$-trajectories and right hand sides, (4.38) can only be solved numerically. The simplest solution is to use a finite difference approximation. However, this leads to rather large systems of equations. For Johnnie and Lola, control cycles of 1.5 . . . 3.5 ms are used, so a trajectory for three 1 s steps requires solving a system of up to 1500 linear equations. Using the proposed collocation method, a good approximate solution for a 3 s trajectory can be obtained using only 30 equally spaced control points. RUSSEL and SHAMPINE give the following description of the collocation method:

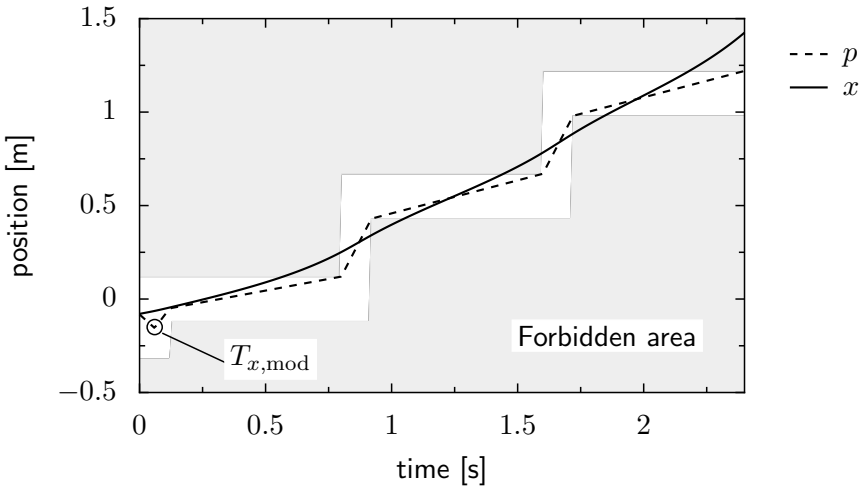


Figure 4.18: Example solution of analytical method for acceleration from 20 cm to 55 cm steps. The CoG position x is drawn with a solid, the ZMP position p with a dashed line. Areas outside of the support polygon are shown in gray.

> "[The method involves] forming an approximate solution as a linear combination of a convenient set of functions, the coefficients of which are determined by requiring the combination to satisfy the differential equation at certain points [...]" [104]

We use a cubic spline base, i.e., piecewise defined polynomials with continuity conditions built in. Consequently, two small instead of one large system of equations must be solved, which further decreases the numerical cost [104].

A cubic spline $s(t)$ on the interval $t \in [t_0, t_{n-1}]$ consists of a set of third order polynomials $s_i(t)$ defined on $t \in [t_i, t_{i+1}]$, where t_i define a partition of $[t_0, t_{n-1}]$. An important property of cubic splines is their linearity in the control points $\boldsymbol{p}$. Denoting the gradient of $\boldsymbol{x}$ with respect to $\boldsymbol{y}$ by $\nabla_{\boldsymbol{y}} \boldsymbol{x}$, the polynomial for $t \in [t_i, t_{i+1}]$ can be written as:

$$s_i(t) = (\nabla_{\boldsymbol{p}}\, s_i(t))\, \boldsymbol{p} \tag{4.45}$$
$$\dot{s}_i(t) = (\nabla_{\boldsymbol{p}}\, \dot{s}_i(t))\, \boldsymbol{p} \tag{4.46}$$
$$\ddot{s}_i(t) = (\nabla_{\boldsymbol{p}}\, \ddot{s}_i(t))\, \boldsymbol{p} \tag{4.47}$$

The matrices $\nabla_{\boldsymbol{p}}\, s_i(t), \nabla_{\boldsymbol{p}}\, \dot{s}_i(t)$ and $\nabla_{\boldsymbol{p}}\, \ddot{s}_i(t)$ are independent of $\boldsymbol{p}$. Calculating the gradients involves solving a tridiagonal system of linear equations, which can be done very efficiently by LR-decomposition requiring only $\mathrm{O}(n-2)$ operations. Relevant equations for calculating cubic spline gradients are listed in Appendix E.

Defining $\eta \approx y_b$ as the approximate solution to (4.38) and choosing the t_i as collocation points yields:

$$m_b \left(z_{b,i} \left(\nabla_{\boldsymbol{p}}\, \ddot{\eta}_i\right) - \left(\nabla_{\boldsymbol{p}}\, \eta_i\right) \left(\ddot{z}_{b,i} + g\right)\right) \boldsymbol{p} = -\tilde{T}_{x,i}, \quad i \in [0, n-1]. \tag{4.48}$$

That is, there are n equations for the n spline control points p_i. To simplify calculations, we use natural splines, i.e., accelerations are zero at the boundaries by definition. To be able to represent non-zero accelerations, we introduce one virtual

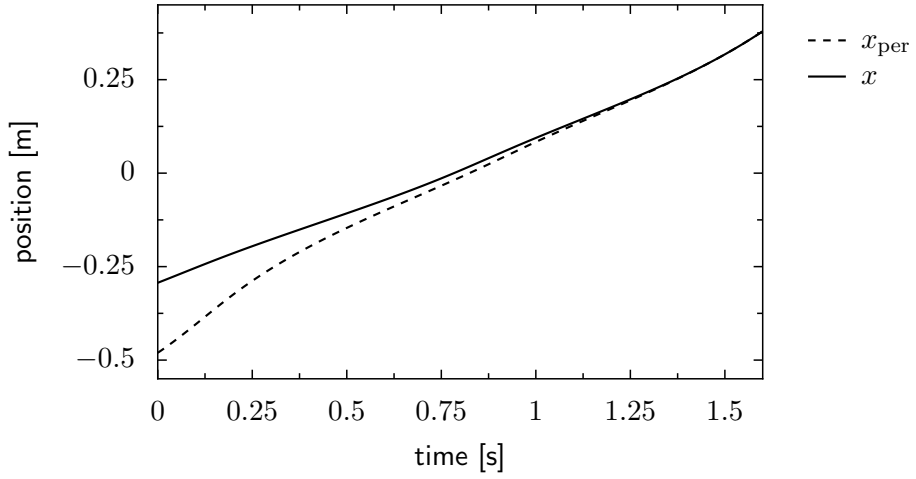


Figure 4.19: Center of gravity trajectory calculated using the proposed collocation method. The dashed line is the periodic reference trajectory x_{ref} and the solid line the resulting trajectory x for the next two steps.

control point before t_0 and one after t_{n-1}. Thus, (4.48) leaves the two virtual control points as free parameters, as would be expected for a second order ODE. In order to satisfy all boundary conditions, two additional degrees of freedom are required. We propose modifying the reference moment during the double support phases of the first and second step, such that:

$$T_x(t) = T_{x,ref}(t) + \gamma_0 \Delta T_{x,0}(t) + \gamma_1 \Delta T_{x,1}(t) \tag{4.49}$$

Here $\Delta T_{x,i}$ are shape functions and γ_i are the additional free parameters. In the default implementation, $\Delta T_{x,i}$ are trapezoids with unit height that span the i-th step. Thus, the full set of equations is:

$$\left(m_b \left(z_b \nabla_{\boldsymbol{p}} \ddot{\eta} - \nabla_{\boldsymbol{p}} \eta (\ddot{z}_b + g)\right) \boldsymbol{p}\right)\big|_{t=t_i} = -\left(\tilde{T}_x + \sum_{k=0}^{1} \gamma_k \Delta T_{x,k}\right)\Bigg|_{t=t_i} \quad i \in [1, n-2] \tag{4.50}$$

$$\nabla_{\boldsymbol{p}} \eta_1 \boldsymbol{p} = y_{\text{cur}}(t_{\text{step},0}) \tag{4.51}$$

$$\nabla_{\boldsymbol{p}} \dot{\eta}_1 \boldsymbol{p} = \dot{y}_{\text{cur}}(t_{\text{step},0}) \tag{4.52}$$

$$\nabla_{\boldsymbol{p}} \eta_{n-2} \boldsymbol{p} = y_{\text{per}}(t_{\text{step},2}) \tag{4.53}$$

$$\nabla_{\boldsymbol{p}} \dot{\eta}_{n-2} \boldsymbol{p} = \dot{y}_{\text{per}}(t_{\text{step},2}) \tag{4.54}$$

Note that indices $i = 0, n-1$ are omitted in (4.50), since p_0, p_{n-1} are virtual control points.

Figure 4.19 shows a solution obtained by the proposed method while accelerating from 30 cm to 56 cm steps. The trajectory converges towards the periodic reference (dashed line). Modifications to the original ZMP reference are shown in Figure 4.20.

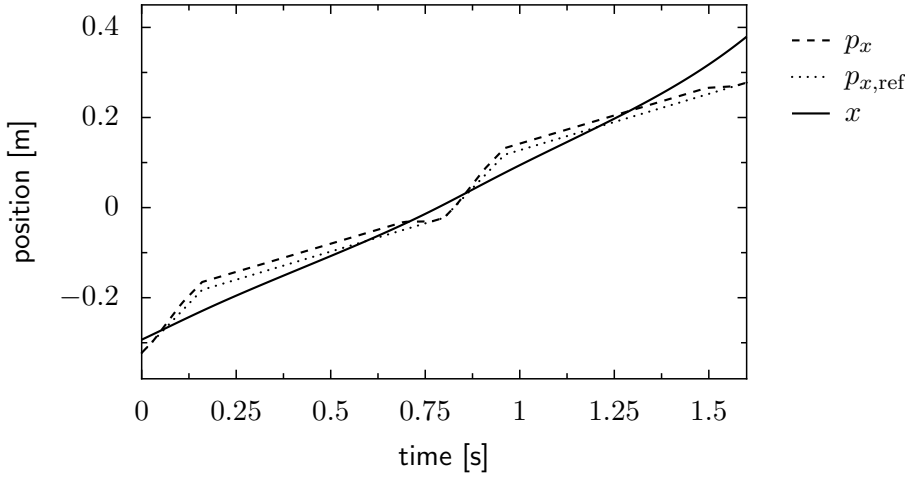


Figure 4.20: Center of gravity trajectory (x), reference ($p_{x,\mathrm{ref}}$) and modified ZMP trajectories (p_x).

Discussion

Both methods presented generate feasible walking patterns. However, the collocation method is more general, since it can handle complex moment reference patterns, account for swing leg motion and vertical motion of the CoG. The benefit of using the three point mass model, which becomes possible using the collocation method, is visible from Figure 4.17, which shows a reduction of the torso swaying motion by more than 45 % in sagittal direction. Of course, the more powerful solution method comes at a certain cost. Table 4.3 shows calculation times for both methods using Lola's onboard PC (Intel Core 2 Duo T7600, 2.33 GHz). Calculation times for the analytical method are negligible compared to other calculation such as inverse kinematics. While the collocation method is slower, it is still sufficiently fast to allow planning to complete within less than one control cycle.

4.7 Camera Head Control

Lola's camera head has three degrees of freedom: pan, tilt and camera vergence. It can be operated in two modes: either target joint angles are entered by a human operator and the head executes a point-to-point trajectory, or the robot fixates a 3-D point in space, taking the kinematics of the entire robot into account. The first mode is useful for walking experiments without computer vision, while the second mode is used during autonomous locomotion. In the latter case, the computer vision system determines the 3-D point P the robot should look at (cf. Chapter 6). In the following, the second mode of operation will be presented.

4.7.1 Inverse Kinematics

Figure 4.21 illustrates the kinematics of Lola's camera head. Ideally, the reference point P should be at the center of both camera images, i.e., P should coincide with the intersection of the optical axes, called the fixation point. Since the head has three

Table 4.3: Calculation times for analytical solution and by spline collocation for 20 walking steps

Method	Max [ms]	Min [ms]	Average [ms]
Analytical solution	0.063	0.043	0.052
Collocation with 40 points	1.039	0.973	0.985
Collocation with 20 points	0.415	0.388	0.396

degrees of freedom, this can be achieved by controlling only the head joints, while the remaining generalized coordinates are assumed to be given. Since the actual robot pose is used when calculating ideal head joint angles, the robot automatically compensates for self-motion and disturbances and inertially stabilizes the cameras. The fixation point P relative to the current planning frame I_i can be written as:

$$ {}_{I_i}\boldsymbol{r}_{I_iP} = \underbrace{\boldsymbol{A}_{I_iI_0}}_{\text{odometry}} \Big(\underbrace{{}_{I_0}\boldsymbol{r}_P}_{\text{fixation point}} - \underbrace{{}_{I_0}\boldsymbol{r}_{I_0I_i}}_{\text{odometry}} \Big) \tag{4.55} $$

The position and orientation of the current stance foot relative to the first reference frame I_0 are determined by odometry, while the fixation point ${}_{I_0}\boldsymbol{r}_P$ is given by the computer vision system. The vector from the current planning frame to the intersection C of pan and tilt axes (${}_{I_i}\boldsymbol{r}_C$) is calculated using IMU and encoder data. The vector from C to P in the upper body planning frame T (${}_T\boldsymbol{r}_{CP}$) is then given by:

$$ {}_T\boldsymbol{r}_{CP} = \boldsymbol{A}_{TI_i}({}_{I_i}\boldsymbol{r}_{I_iP} - {}_{I_i}\boldsymbol{r}_C) \tag{4.56} $$

From this, pan and tilt angles can be calculated using elementary geometry:

$$ \theta_{\text{pan}} = -\operatorname{atan2}\left({}_T\boldsymbol{r}_{CP,z}\,, {}_T\boldsymbol{r}_{CP,x}\right) \tag{4.57} $$

$$ \theta_{\text{tilt}} = \operatorname{atan2}\left({}_T\boldsymbol{r}_{CP,y}\,, \sqrt{{}_T\boldsymbol{r}_{CP,x}^2 + {}_T\boldsymbol{r}_{CP,z}^2}\right) \tag{4.58} $$

For given pan and tilt angles, the vergence angle θ_{verg} can be calculated as:

$$ \theta_{\text{verg}} = 2\operatorname{atan2}\left(\frac{\overline{F_LF_R}}{2}, \overline{C'P}\right) \tag{4.59} $$

Here $\overline{F_LF_R}$ is the distance between the right (F_R) and left (F_L) focal points and $\overline{C'P}$ the distance between the midpoint C' between the focal points and P. If $\overline{C'P} \gg \overline{F_LF_R}$, which is always the case for autonomous locomotion, we can simplify the calculations by setting $\overline{C'P} \approx \overline{CP}$. Using this approximation, the vergence angle becomes independent of pan and tilt angles. Ideal angular velocities $\dot{\theta}_{\text{pan}}, \dot{\theta}_{\text{tilt}}$ and $\dot{\theta}_{\text{verg}}$ are calculated in a similar fashion as joint angles.

4.7.2 Reference Trajectory Generation

The trajectory of the viewing target can be discontinuous, e.g., when attention switches to a different object. Therefore, a simple tracking controller would have

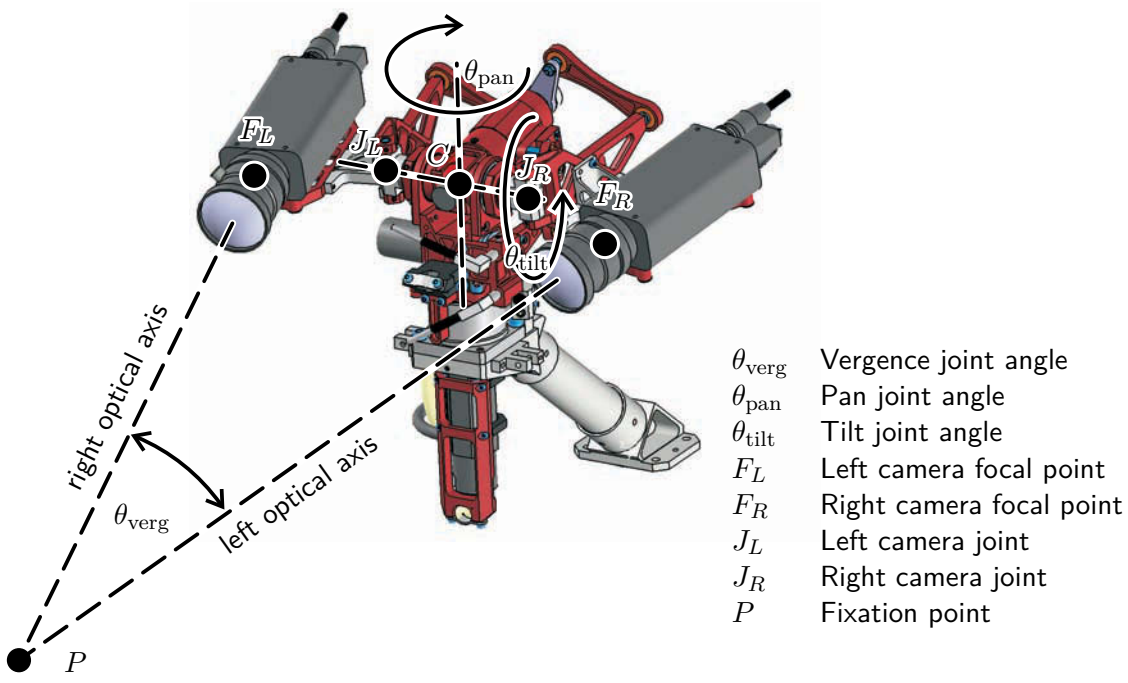


Figure 4.21: Kinematics of Lola's three degree of freedom camera head

to use very low gains to avoid damaging the robot. This, however, would lead to unacceptable performance when tracking an object.

To solve this problem, a two-part control strategy is adopted that distinguishes between tracking an object or moving to a new viewing target. This enables generation of smooth trajectories for the joint position controller while achieving good tracking of stationary or smoothly moving objects.

The strategy was designed from a control systems point of view, but there is a remarkable similarity to eye movements in biological systems. Here there also is a distinction between volitional saccades on the one hand and smooth pursuit or optokinesis on the other [56].

While smooth pursuit is often studied in the context of moving objects, some authors have suggested that it evolved to keep the point of interest at the center of the retina during self motion [56], which is also the main purpose of this control module for the robot. In contrast to biological systems, however, Lola uses no direct visual feedback in the camera motion control loop.

Switching is based on the current tracking error of the viewing target: for small tracking errors the smooth pursuit controller is selected and a saccade is initiated otherwise.

The tracking controller calculates target angles for the joint position controller by filtering the ideal target trajectories obtained by inverse kinematics. The controller is implemented as a second order low-pass filter with rate limiter that outputs desired position, velocity and acceleration.

Trajectories for moving towards a new viewing target (saccades) are parameterized as fifth-order polynomials. The six parameters are determined from position, velocity and acceleration at the current control cycle and at the end of the trajectory. The final acceleration is set to zero, while the velocity and position are set to those obtained

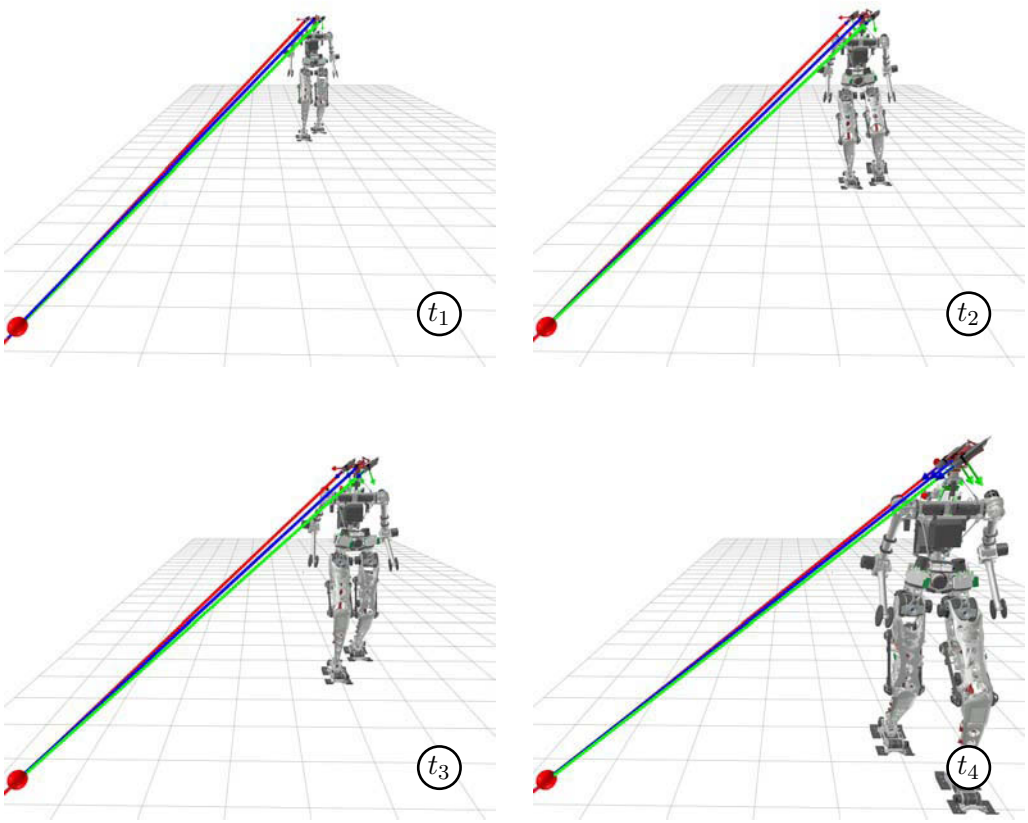


Figure 4.22: Lola uses odometry and the full kinematic model to keep a 3-D point of interest (shown as a red sphere) at the center of both camera images. Optical axis of right and left camera (respectively shown in red and green) intersect at the target point.

from inverse kinematics. The duration of the trajectory is calculated numerically from the maximal admissible speed for the saccadic motion.

The head trajectory is recalculated every control cycle, so the viewing target can be switched at any time. Figure 4.22 illustrates the head movement when tracking a static point P. Figure 4.23 shows images taken with Lola's left camera during a walking experiment. The computer vision system was deactivated and Lola was controlled using a joystick. The viewing target is kept at the center of the camera image using only odometry, angular sensors and inertial measurements.

4.8 Contact Force Distribution

The planning methods for center of gravity and contact force trajectories presented in the previous section only plan the total contact force acting on both feet. During the double support phase, the total contact force $\boldsymbol{F}_{\text{tot}}$ and contact moment $\boldsymbol{T}_{\text{tot}}$ is given by:

$$\boldsymbol{F}_{\text{tot}} = \boldsymbol{F}_0 + \boldsymbol{F}_1 \tag{4.60}$$

$$\boldsymbol{T}_{\text{tot}} = \boldsymbol{T}_0 + \boldsymbol{T}_1 + \boldsymbol{\Delta r}_{\text{swing}} \times \boldsymbol{F}_{\text{swing}} \tag{4.61}$$

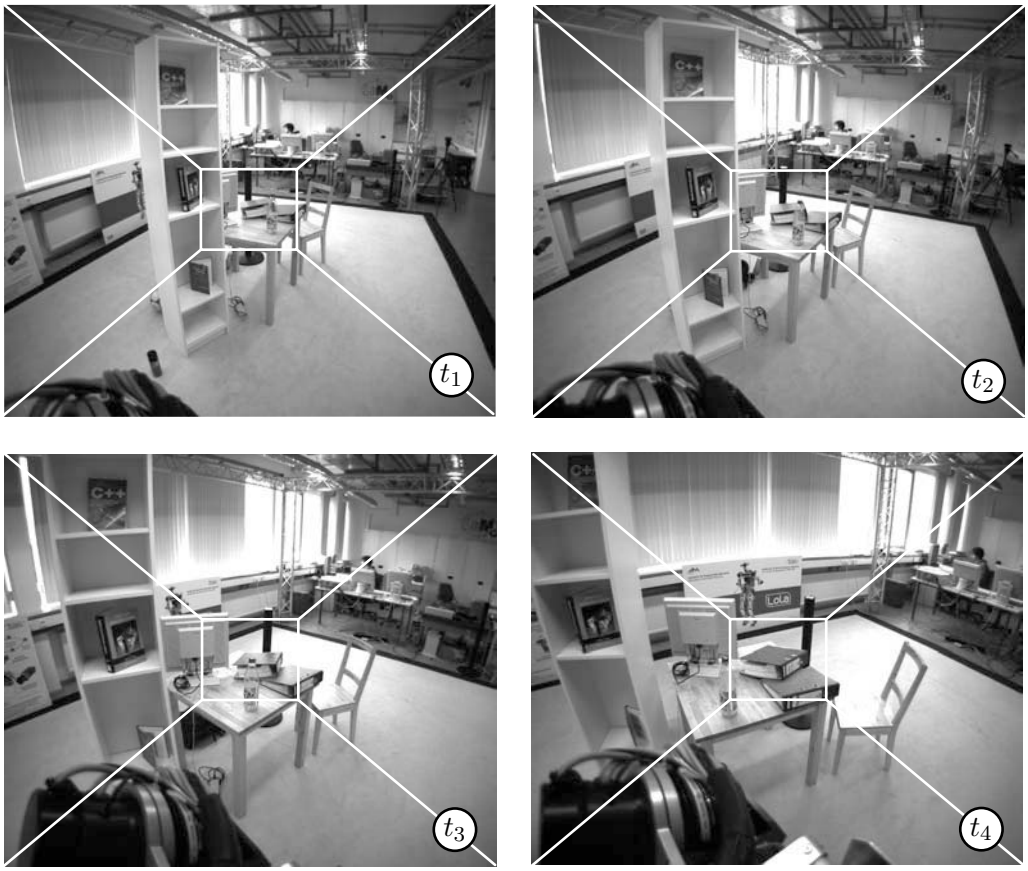


Figure 4.23: Camera images taken during a walking experiment. The static fixation point is located approximately at the center of the binder.

Here $\boldsymbol{F}_i, \boldsymbol{T}_i$ denote the contact force and moment acting on the i-th foot and $\boldsymbol{\Delta r}_{\text{swing}}$ is the vector from stance to swing foot reference point. The force distribution between left and right foot is defined by the load factors ρ_i:

$$\rho_i = \frac{F_{i,z}}{F_{0,z} + F_{1,z}} \tag{4.62}$$

Here $F_{i,z}$ is the vertical component of the contact force acting on the i-th foot. The same load factors are used for all force components, leading to the following equations:

$$\boldsymbol{F}_i = \rho_i \boldsymbol{F}_{\text{total}} \tag{4.63}$$

$$\boldsymbol{T}_i = \rho_i \left(\boldsymbol{T}_{\text{total}} - \boldsymbol{\Delta r}_{\text{swing}} \times \boldsymbol{F}_{\text{swing}} \right) \tag{4.64}$$

During the single support phase, $\rho = 1$ for the stance leg and $\rho = 0$ for the swing leg. A linear interpolation between these two values is used during the double support phase. The corner that would result at the beginning and end of the single support phases when using a simple linear interpolation is smoothed using a fifth-order polynomial. Figure 4.24 shows the desired and actual load factors for Lola during periodic walking. Interestingly, load factors for humans show a very similar pattern.

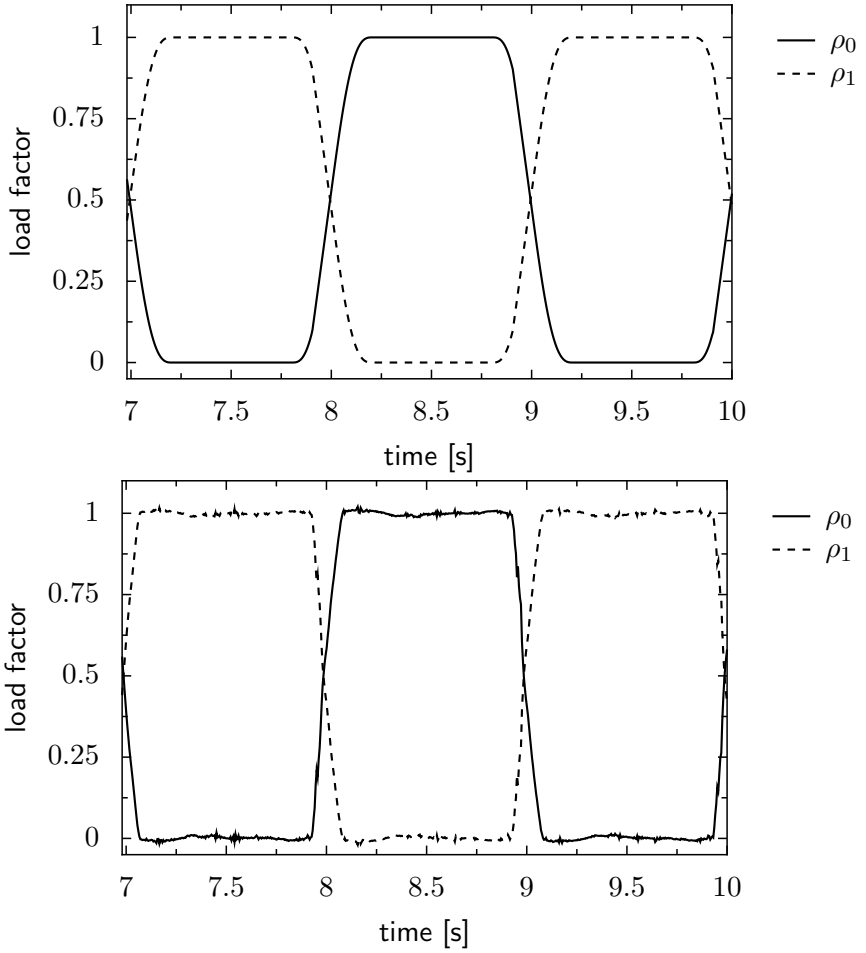


Figure 4.24: Desired (top) and actual load factors (bottom) for Lola walking in a dynamics simulation

Figure 4.25 shows load patterns obtained from force plate measurements at a walking speed of $\approx 1.8\,\mathrm{m/s}$.[2]

4.9 Additional Components in Trajectory Generation

The basic and most important parts of the trajectory generation system were presented in the preceding sections. This section gives an overview of additional components that improve walking stability and smoothness, especially for faster walking.

Compensating Foot Compliance

The foot trajectories described above satisfy the kinematic constraints when the foot is not deformed. However, this leads to dynamically inconsistent trajectory definitions: if the foot is undeformed, the contact forces must be zero, but non-zero

2 Experiments were performed at the Locomotion Laboratory, Institute of Sport Science, Friedrich Schiller Universität Jena. The author would like to thank Susanne Lipfert for kindly providing the data.

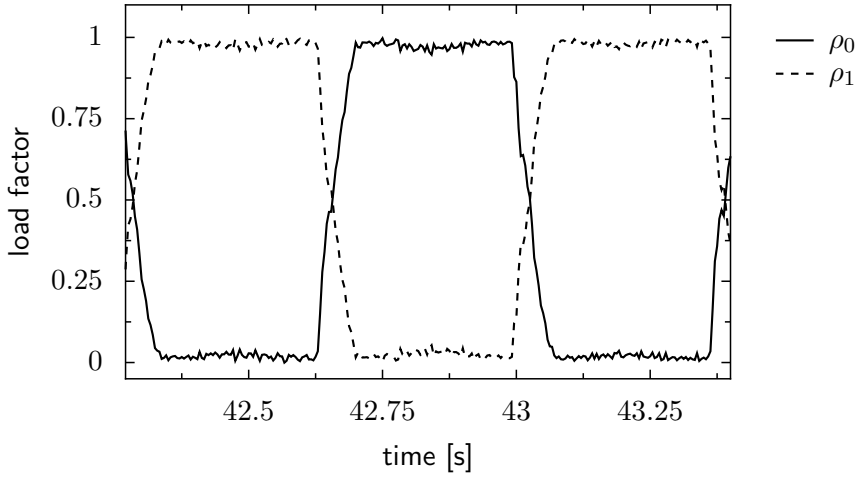


Figure 4.25: Measured load factors for a human walking on a treadmill

contact forces are necessary during walking. Two methods of dealing with this problem were implemented. In the first approach, the foot trajectories are modified, such that the deformation required for the desired contact forces is included. In the second approach, this compensation is performed by the contact force controller (cf. Section 5.3).

Compensating Torso Inclination

The foot trajectories are controlled in the upper body planning frame. Consequently, an error in the upper body inclination can lead to early or late ground contact of the swing foot. This can be compensated by adding a small modification to the vertical swing foot position and rotation. The modification term is returned to zero during the stance phase.

Arm Swing

Especially at higher walking speeds, fast leg motion can lead to high vertical contact moments, which in turn cause the robot to slip. This moment can be reduced by swinging the arms back and forth opposite to the leg motion. For Lola, the arm CoG is controlled relative to the robot CoG. The reference values are calculated from the position of the CoG of the opposite leg relative to the robot CoG. Since the arms are shorter than the legs, the compensating motion must be scaled down accordingly.

4.10 Chapter Summary

In Chapter 4 we presented the system for real-time gait planning and trajectory generation. The desired walking speed and direction, as well as a viewing target are obtained either from a computer vision system or a human operator. From these inputs, ideal motion patterns are generated in a top-down fashion, coordinated by a finite state machine. At the upper hierarchical levels sequences of walking steps and the length of different phases of gait are planned. Step sequence planning includes a reactive, model-based strategy for stepping over obstacles using a local environment map.

A detailed explanation of coordinate systems and the spatial orientation representation used to describe task-space trajectories was also given.

A novel method for generating feasible center of gravity trajectories using a simplified multi-mass model was presented. It is based on cubic spline collocation and can generate trajectories for which the height of the center of gravity is not constant.

Control of the three degree of freedom camera head is fully integrated into the walking control. Reference trajectories are generated from a given 3-D fixation point, taking the entire robot kinematics into account. This novel approach allows the robot to compensate self-motion and to inertially stabilize the cameras. To the author's knowledge, this is the first time gaze control was fully integrated into a humanoid robot's full-body control, in order to compensate self-motion and disturbances and to inertially stabilize the camera images.

5 Feedback Control

5.1 Introduction

5.1.1 Overview

For large bipeds such as Lola and Johnnie, simply tracking planned reference trajectories does not lead to stable walking. Modeling errors of robot and environment, as well as disturbances quickly lead to instability of the unactuated degrees of freedom. Therefore, the reference trajectories must be adapted on-line in order to stabilize the robot. The feedback control system can be divided into four main components:

- Contact force modification
- Hybrid position/force control
- Redundancy resolution and inverse kinematics
- Joint position control

Special emphasis is put on designing a general control method applicable to robots with a wide range of different kinematics and mass distributions. Figure 4.1 illustrates the structure of the walking control system with trajectory generation and feedback control. The model-based approach presented in this chapter has been successfully applied to both Johnnie and Lola in simulations and experiments.

5.1.2 Background and Related Work

Stabilizing Control of Biped Robots

The most common method of biped walking control is to first calculate theoretically stable walking patterns either on-line [6, 45] or off-line [9]. The trajectory is then modified on-line using feedback control in order to achieve stable walking. A large number of methods for stabilizing control of bipeds have been proposed and a number of similar ideas have been successfully implemented on human-sized humanoid robots. In the following, a brief review of the approaches most relevant to this thesis is given.

A common strategy for stabilizing the upper body inclination is to control the contact moments at the feet [44, 61, 119, 124]. Usually, the foot moments are measured by six-axis force/torque sensors and controlled via position control of the ankle joints. Another common strategy is to accelerate the center of gravity, creating a reaction force that stabilizes the robot [74, 116].

An interesting strategy for stabilizing planar bipeds based on methods for torque-limited control of robot manipulators [92] was proposed in [13]. By using time scaling of the trajectory, both the geometric path and the CoP can be tracked. However,

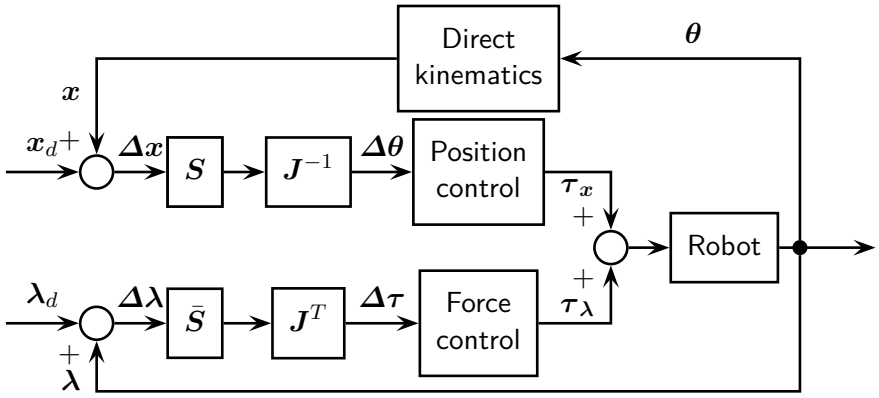


Figure 5.1: Hybrid position/force control for torque controlled device, as proposed in [15]. $\boldsymbol{S}, \bar{\boldsymbol{S}}$ denote respectively the selection matrices for position and force controlled dimensions, $\boldsymbol{\lambda}$ is the contact force and $\boldsymbol{x}$ the task-space position.

since time scaling only adds one additional control input, application to 3-D robots is not straightforward.

In order to reduce landing impacts, many robots also incorporate an active control of vertical contact forces. This can be achieved by measuring the contact force and changing position set points [44, 61]. Another strategy involves reducing joint position control gains during initial contact [73, 116], i.e., using indirect force control.

Fujimoto proposed a hybrid position/force control scheme based on task-space position control [23]. Contact moments in the ground plane are not used for stabilization, but for tracking a reference CoP instead. The remaining contact forces are used for controlling the robot's pose and torso height. Contrary to the method proposed by the author, stabilization of the CoG trajectory is achieved by modifying the swing leg trajectory according to Kajita's *Linear Inverted Pendulum Mode* (LIPM) [43].

Löffler proposed an approach based on feedback linearization for the biped Johnnie. Reportedly, the performance of this method was limited by the available sensor bandwidth, computational power and accuracy of the models. In experiments, the performance of a method based on impedance control proved to be superior [61]. The author believes that the second method was superior, because contact moments were always actively controlled to stabilize the upper body. In the first method based on feedback linearization, contact forces were treated as external forces to be compensated as long as they did not reach the limit of admissible forces. In that case the system switched to tracking the contact force to avoid a rotation of the foot along its edge. Therefore, the basic structure of controlling contact forces to stabilize the robot is maintained in the control system proposed in this thesis.

Hybrid Position/Force Control

Hybrid position/force control was originally proposed by Craig and Raibert for interaction control of robot manipulators [15]. The basic approach is to partition the task-space into force and position controlled sub-spaces according to the geometry of the assembly or manipulation task. This allows precise motion control in unconstrained directions while force control allows the robot to adapt its motion to the environment's geometry and stiffness.

In [15] the individual joint control terms are a linear combination of contributions

from a PID-type position control and direct force control, as illustrated in Figure 5.1. More sophisticated approaches using dynamics models of the manipulator and nonlinear decoupling were subsequently proposed by Khatib [55] and others.

Classically, hybrid position/force control was formulated for torque controlled manipulators. However, for robots with high structural stiffness, actuated by electric motors through reduction gears with relatively high ratios (≥ 50), such as Johnnie and Lola, very low tracking errors can be achieved using simple decoupled joint position control. Furthermore, direct joint position control enables good disturbance rejection and compensation of gear friction with high bandwidth [2]. This has led to the development of hybrid position/force control based on inner position or velocity control loops [110, Ch. 7.4.3]. This approach is pursued in the following.

5.2 Contact Force Modification

The dynamics of a position controlled biped exhibit certain structural properties that can be exploited in the stabilizing control. As detailed in Section 2.6.1, pp. 41 to 43, the joint angles approximately follow the reference trajectories, while the upper body DoFs are not directly actuated. The equations for this reduced model are given by:

$$\begin{pmatrix} \boldsymbol{M}_{TT} & \boldsymbol{M}_{TJ} \\ \boldsymbol{M}_{JT} & \boldsymbol{M}_{JJ} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}}_T \\ \ddot{\boldsymbol{q}}_J \end{pmatrix} + \begin{pmatrix} \boldsymbol{h}_T \\ \boldsymbol{h}_J \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\tau} \end{pmatrix} + \begin{pmatrix} \boldsymbol{J}_{\lambda,T} \\ \boldsymbol{J}_{\lambda,J} \end{pmatrix} \boldsymbol{\lambda} \tag{5.1}$$

Here $\boldsymbol{J}_\lambda$ is the Jacobian projecting the contact forces $\boldsymbol{\lambda}$ into the space of generalized coordinates. Since the joint angles $\boldsymbol{q}_J$ are tracked accurately, only the first line is relevant for the stabilizing controller. Assuming the robot does not slip and the tangential contact stiffness is high, the number of unactuated DoFs is further decreased from six to three: the vertical CoG position and the upper body inclination in the sagittal and lateral planes. Note that errors in the CoG and upper body position are a function of the upper body inclination error. Consequently, controlling the upper body inclination error and the CoG error are equivalent.

Equation (5.1) shows that while the upper body DoFs are not directly actuated, they can be controlled via the contact forces $\boldsymbol{\lambda}$. Moreover, according to the basic laws of mechanics, the overall linear and angular momentum can *only* be influenced via the contact forces. Stabilizing control is therefore based on modifying the contact forces in order to control the upper body inclination error and the height of the center of gravity.

In a first step, additional forces and moments that must act at the robot's CoG are calculated using a PD-type control law:

$$\Delta T_x^{CoG} = K_{P,x} \Delta\varphi_x + K_{D,x} \Delta\dot{\varphi}_x \tag{5.2}$$

$$\Delta T_y^{CoG} = K_{P,y} \Delta\varphi_y + K_{D,y} \Delta\dot{\varphi}_y \tag{5.3}$$

$$\Delta F_z^{CoG} = K_{P,z} \Delta z_{CoG} + K_{D,z} \Delta\dot{z}_{CoG} \tag{5.4}$$

The remaining components of $\Delta\boldsymbol{F}^{CoG}$ and $\Delta\boldsymbol{T}^{CoG}$ are set to zero. Denoting the ideal force references calculated during trajectory generation as ${}_I\boldsymbol{F}_{d0}^{CoG}$ and ${}_I\boldsymbol{T}_{d0}^{CoG}$,

the modified total contact force is given by:

$$
\begin{aligned}
{}_I\boldsymbol{F}_{d1} &= {}_I\boldsymbol{F}_{d0}^{CoG} + {}_I\boldsymbol{\Delta F}^{CoG} && (5.5)\\
{}_I\boldsymbol{T}_{d1} &= {}_I\boldsymbol{T}_{d0}^{CoG} + {}_I\boldsymbol{\Delta T}^{CoG} + ({}_I\boldsymbol{r}_{CoG} - {}_I\boldsymbol{r}_{\mathrm{ref}}) \times {}_I\boldsymbol{F}_{d1}^{CoG} && (5.6)
\end{aligned}
$$

The reference point for the total contact force and moment ${}_I\boldsymbol{r}_{\mathrm{ref}}$ is the sum of the force/torque sensor positions ${}_I\boldsymbol{r}_{FTS,i}$ weighted with the corresponding load factor ρ_i (cf. (4.62)):

$$
{}_I\boldsymbol{r}_{\mathrm{ref}} = \rho_0\, {}_I\boldsymbol{r}_{FTS,0} + \rho_1\, {}_I\boldsymbol{r}_{FTS,1} \tag{5.7}
$$

Denoting the vector from the i-th force/torque sensor to the reference point by $\boldsymbol{\Delta r}_{\mathrm{ref},i}$, the total contact force can be written as:

$$
\begin{aligned}
\boldsymbol{F}_{\mathrm{tot}} &= \boldsymbol{F}_0 + \boldsymbol{F}_1 && (5.8)\\
\boldsymbol{T}_{\mathrm{tot}} &= \boldsymbol{T}_0 + \boldsymbol{T}_1 + \boldsymbol{\Delta r}_{\mathrm{ref},0} \times \boldsymbol{F}_0 + \boldsymbol{\Delta r}_{\mathrm{ref},1} \times \boldsymbol{F}_1 && (5.9)
\end{aligned}
$$

Here $\boldsymbol{F}_i, \boldsymbol{T}_i$ are the force and moment acting on the i-th foot. The sum of vertical contact forces $F_{z,0} + F_{z,1}$ is given by (5.8), but the difference can be chosen within the bounds admissible by the unilateral contact. Especially for longer steps, modifying this difference is an effective means of contributing to $\boldsymbol{T}_{\mathrm{tot}}$. Since the contact moments are limited by the size of the feet, the author proposes to maximize the contribution of normal forces to $\boldsymbol{T}_{\mathrm{tot}}$ in order to increase the moment available for stabilization.

The contribution to the horizontal contact moment obtained by adding/subtracting $\Delta F_{z,\mathrm{stab}}$ to the vertical component of left and right contact forces is given by:

$$
\begin{aligned}
\boldsymbol{\Delta T}_N &= \boldsymbol{\Delta r}_{\mathrm{ref},0} \times \boldsymbol{e}_z \Delta F_{z,\mathrm{stab}} + \boldsymbol{\Delta r}_{\mathrm{ref},1} \times \boldsymbol{e}_z(-\Delta F_{z,\mathrm{stab}})\\
&= \Delta F_{z,\mathrm{stab}} \begin{pmatrix} \Delta r_{\mathrm{ref},0,y} - \Delta r_{\mathrm{ref},1,y} \\ -(\Delta r_{\mathrm{ref},0,x} - \Delta r_{\mathrm{ref},1,x}) \\ 0 \end{pmatrix}\\
&= \Delta F_{z,\mathrm{stab}} \begin{pmatrix} \Delta r_{\mathrm{ref},01,y} \\ -\Delta r_{\mathrm{ref},01,x} \\ 0 \end{pmatrix}
\end{aligned} \tag{5.10}
$$

That is, the additional contact moment is orthogonal to the connecting line between left and right force sensor frame. Therefore, the contribution due to $\Delta F_{z,\mathrm{stab}}$ is calculated in a rotated coordinate system N whose y-axis is aligned with the connecting line between left and right force sensor. The total contact moment in the rotated frame is:

$$
{}_N\boldsymbol{T}_{\mathrm{tot}} = \underbrace{\frac{1}{\sqrt{\Delta r_{\mathrm{ref},01,x}^2 + \Delta r_{\mathrm{ref},01,y}^2}} \begin{pmatrix} \Delta r_{\mathrm{ref},01,y} & -\Delta r_{\mathrm{ref},01,x} & 0 \\ \Delta r_{\mathrm{ref},01,x} & \Delta r_{\mathrm{ref},01,y} & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\text{rotation matrix } \boldsymbol{A}_{NI}} {}_I\boldsymbol{T}_{\mathrm{tot}} \tag{5.11}
$$

Modifying the normal force adds

$$\sqrt{\Delta r_{\mathrm{ref},01,x}^2 + \Delta r_{\mathrm{ref},01,y}^2}\Delta F_{z,\mathrm{stab}}$$

to the x-component of total contact moment in the coordinate frame N.

Normal forces must be positive and the swing leg contact force must be zero. Additionally, the normal contact force is limited to $\rho_i F_{z,\max}$, where $F_{z,\max}$ is a bound for the maximum normal force. This leads to the following limits for $\Delta F_{z,\mathrm{stab}}$:

$$\begin{aligned} \Delta F_{z,\mathrm{stab}} &\geq \Delta F_{z,\mathrm{stab,min}} &&= \max\left(-\rho_0 F_{\mathrm{tot},z}, \rho_1(F_{\mathrm{tot},z} - F_{z,\max})\right) \\ \Delta F_{z,\mathrm{stab}} &\leq \Delta F_{z,\mathrm{stab,max}} &&= \min\left(\rho_0(F_{z,\max} - F_{\mathrm{tot},z}), \rho_1 F_{\mathrm{tot},z}\right) \end{aligned} \tag{5.12}$$

A typical value for $F_{z,\max}$ is twice the robot's weight. The admissible normal force modification that minimizes the necessary contact moments is then given by:

$$\Delta F_{z,\mathrm{stab}}^{\star} = \frac{{}_N T_{\mathrm{tot},x}}{\sqrt{\Delta r_{\mathrm{ref},01,x}{}^2 + \Delta r_{\mathrm{ref},01,y}{}^2}} \tag{5.13}$$

$$\Delta F_{z,\mathrm{stab}} = \begin{cases} \Delta F_{z,\mathrm{stab,max}} & \text{if} \quad \Delta F_{z,\mathrm{stab}}^{\star} > \Delta F_{z,\mathrm{stab,max}} \\ \Delta F_{z,\mathrm{stab,min}} & \text{if} \quad \Delta F_{z,\mathrm{stab}}^{\star} < \Delta F_{z,\mathrm{stab,min}} \\ \Delta F_{z,\mathrm{stab}}^{\star} & \text{else} \end{cases} \tag{5.14}$$

Thus, the desired contact forces are calculated as follows:

$${}_I\boldsymbol{F}_0 = \rho_0\, {}_I\boldsymbol{F}_{\mathrm{tot}} + \boldsymbol{e}_z \Delta F_{z,\mathrm{stab}} \tag{5.15}$$

$${}_I\boldsymbol{F}_1 = \rho_1\, {}_I\boldsymbol{F}_{\mathrm{tot}} - \boldsymbol{e}_z \Delta F_{z,\mathrm{stab}} \tag{5.16}$$

$${}_I\boldsymbol{T}_0 = \rho_0\, {}_I\boldsymbol{\Delta} T_{\mathrm{tot}}^{\star} \tag{5.17}$$

$${}_I\boldsymbol{T}_1 = \rho_1\, {}_I\boldsymbol{\Delta} T_{\mathrm{tot}}^{\star} \tag{5.18}$$

$${}_I\boldsymbol{\Delta} T_{\mathrm{tot}}^{\star} = \left[\boldsymbol{A}_{IN}\left({}_N\boldsymbol{T}_{\mathrm{tot}} - \boldsymbol{e}_x \Delta F_{z,\mathrm{stab}} \sqrt{\Delta r_{\mathrm{ref},01,x}{}^2 + \Delta r_{\mathrm{ref},01,y}{}^2}\right)\right] \tag{5.19}$$

Finally, the desired contact forces are calculated by taking the rotation of the force sensor reference frames into account:

$${}_{Fi}\boldsymbol{F}_i = \boldsymbol{A}_{FiI}\, {}_I\boldsymbol{F}_i \tag{5.20}$$

$${}_{Fi}\boldsymbol{T}_i = \boldsymbol{A}_{FiI}\, {}_I\boldsymbol{T}_i \tag{5.21}$$

Here the Fi denotes the reference frame of the i-th force/torque sensor. The vector of modified contact forces $\boldsymbol{\lambda}_{d1}$ is then given by:

$$\boldsymbol{\lambda}_{d1}^T = \begin{pmatrix} {}_{Fi}\boldsymbol{F}_0^T & {}_{Fi}\boldsymbol{T}_0^T & {}_{Fi}\boldsymbol{F}_1^T & {}_{Fi}\boldsymbol{T}_1^T \end{pmatrix} \tag{5.22}$$

Figure 5.2 shows results of a type 2 simulation of Lola walking at 2.2 km/h with and without modifying normal forces for controlling the upper body inclination. By modifying the normal forces, the peak upper body inclination error is decreased by more than 50%.

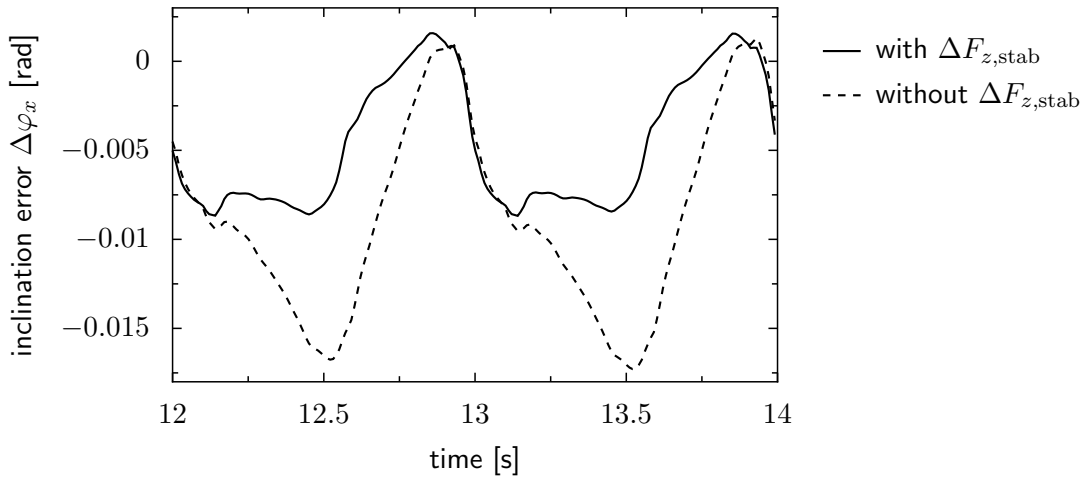


Figure 5.2: Simulation of Lola walking at 2.2 km/h with (solid line) and without (dashed line) using normal forces for upper body stabilization

5.3 Hybrid Position/Force Control

Stabilization of the robot is based on controlling a subset of the contact forces while tracking reference trajectories described in task-space coordinates. To control the contact forces, the reference trajectories are modified along some task-space dimensions. That is, a hybrid position/force control with inner position control loop is used.

An explicit contact model is used to calculate the required trajectory modification for tracking contact forces. This model, illustrated in Figure 5.3, consists of a set of decoupled point contacts with stiffness $\boldsymbol{C}_i$ and negligible damping. By using an explicit contact model, changing foot geometries, contact states and contact stiffness can easily be taken into account, which is useful for Lola during heel or toe contact. A further advantage is the fact that the terms in the force control equations have a clear physical meaning. More importantly, control of individual force components is decoupled. If the foot's "tool center point" and/or the force sensor is not at the center of the foot or the foot's contact stiffness is not homogeneous, controlling only the rotation of the foot relative to the upper body or only the ankle joints will generally alter both the contact moment and the normal contact force. This is avoided by taking the foot geometry, contact stiffness and force sensor location into account.

Using the known contact state, i.e., which contacts are opened or closed, the total force $\boldsymbol{F}_j$ and moment $\boldsymbol{T}_j$ acting on foot j are given by:

$$\begin{pmatrix} \boldsymbol{F}_j \\ \boldsymbol{T}_j \end{pmatrix} = \sum_{i \in I_{c,j}} \begin{pmatrix} {}_j\boldsymbol{C}_i \, \boldsymbol{d}_{c,i} \\ {}_j\boldsymbol{\Delta r}_{c,i} \times {}_j\boldsymbol{C}_i \, \boldsymbol{d}_{c,i} \end{pmatrix} \tag{5.23}$$

Here $I_{c,j}$ denotes the subset of I_c on foot j, $\boldsymbol{d}_{c,i}$ is the deformation of the i-th contact element and ${}_j\boldsymbol{\Delta r}_{c,i}$ is the vector from the force sensor frame to the contact element.

Similar to the use of a binary selection vector by Craig et al. to specify force or position controlled dimensions [15], we use selection matrices to specify subsets of $\boldsymbol{\lambda}$

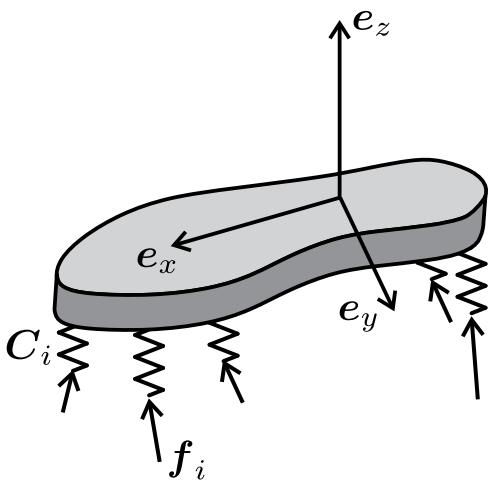


Figure 5.3: Schematic representation of the contact model for real-time stabilizing control

and $\boldsymbol{x}$ to be actively controlled:

$$\boldsymbol{\lambda}_c = \boldsymbol{S}_\lambda \boldsymbol{\lambda} \tag{5.24}$$
$$\boldsymbol{x}_c = \boldsymbol{S}_x \boldsymbol{x} \tag{5.25}$$

Here the subscript c denotes the actively controlled variables and $\boldsymbol{S}_\lambda, \boldsymbol{S}_x$ are binary selection matrices. For this thesis, the forces normal to and the moment in the contact plane are actively controlled.

In the non-position controlled dimensions, we allow a modification of the reference trajectories by $\boldsymbol{\Delta x}_\lambda$ for force control. With $\boldsymbol{K}_\lambda$ as a control gain, we choose the linear error dynamics

$$\boldsymbol{S}_\lambda \left(\boldsymbol{\Delta \dot{\lambda}} + \boldsymbol{K}_\lambda \boldsymbol{\Delta \lambda} \right) = \boldsymbol{0} \tag{5.26}$$

for the actively controlled forces, where $\boldsymbol{\Delta \lambda} = \boldsymbol{\lambda}_{d1} - \boldsymbol{\lambda}$. With $\bar{\boldsymbol{S}}_x$ denoting the complement of $\boldsymbol{S}_x$, we have:

$$\boldsymbol{\Delta \dot{x}}_\lambda = \bar{\boldsymbol{S}}_x \left(\nabla_q \boldsymbol{x} \right) \boldsymbol{\dot{q}}_{\boldsymbol{\Delta \dot{x}_\lambda}} \tag{5.27}$$

From the contact model (5.23) we obtain the relationship:

$$\boldsymbol{\dot{\lambda}} = \left(\nabla_q \boldsymbol{\lambda} \right) \boldsymbol{\dot{q}} \tag{5.28}$$

Substituting (5.28) into (5.26) and rearranging yields:

$$\left(\boldsymbol{S}_\lambda \left(\nabla_q \boldsymbol{\lambda} \right) \right) \boldsymbol{\dot{q}} = \boldsymbol{S}_\lambda \left(\boldsymbol{\dot{\lambda}}_{d1} + \boldsymbol{K}_\lambda \boldsymbol{\Delta \lambda} \right) \tag{5.29}$$

Because of the linear relationship between generalized velocities and workspace velocities, we have $\boldsymbol{\dot{q}} = \boldsymbol{\dot{q}}_{d0} + \boldsymbol{\dot{q}}_{\Delta \dot{\boldsymbol{x}}_\lambda}$, where $\boldsymbol{\dot{q}}_{d0}$ are the ideal generalized velocities corresponding to the ideal task space trajectories $\boldsymbol{x}_{d0}$. Using this relationship and the

Moore-Penrose pseudoinverse $(\cdot)^{\#}$ we can solve for the trajectory modification:

$$\boldsymbol{\Delta\dot{x}}_\lambda = \left[\boldsymbol{S}_\lambda \left(\nabla_q \boldsymbol{\lambda}\right) \left(\bar{\boldsymbol{S}}_x \left(\nabla_q \boldsymbol{x}\right)\right)^{\#}\right]^{\#} \boldsymbol{S}_\lambda \left(\dot{\boldsymbol{\lambda}}_{d1} + \boldsymbol{K}_\lambda \boldsymbol{\Delta\lambda} - \left(\nabla_q \boldsymbol{\lambda}\right) \dot{\boldsymbol{q}}_{d0}\right) \tag{5.30}$$

As detailed in Section 5.2, the desired contact forces $\boldsymbol{\lambda}_{d1}$ are the sum of ideal reference forces $\boldsymbol{\lambda}_{d0}$ and additional forces $\boldsymbol{\Delta\lambda}_{\text{stab}}$ for stabilization. A time derivative for $\boldsymbol{\lambda}_{d0}$ is calculated during trajectory generation, but $\boldsymbol{\Delta\dot{\lambda}}_{\text{stab}}$ is not available since the robot does not measure angular accelerations of the upper body. Therefore, $\dot{\boldsymbol{\lambda}}_{d1} = \dot{\boldsymbol{\lambda}}_{d0}$ is used in the controller implementation.

If the foot deformations are not taken into account during trajectory generation, foot trajectories are planned to not penetrate the environment. This means that for $\boldsymbol{q} = \boldsymbol{q}_{d0}$, the contact element deformations $\boldsymbol{d}_{c,i}$ and the corresponding contact force $\boldsymbol{\lambda}$ vanish and $\left(\nabla_q \boldsymbol{\lambda}\right) \dot{\boldsymbol{q}}_{d0} \equiv 0$. In this case, the trajectory modification due to $\dot{\boldsymbol{\lambda}}_d$ provides a feed-forward compensation of foot deformations. If deformations are taken into account during trajectory planning, we have $\dot{\boldsymbol{\lambda}}_{d0} = \left(\nabla_q \boldsymbol{\lambda}\right) \dot{\boldsymbol{q}}_{d0}$. In both cases, the trajectory generation can be written as:

$$\boldsymbol{\Delta\dot{x}}_\lambda = \left(\boldsymbol{S}_\lambda \left(\nabla_q \boldsymbol{\lambda}\right) \left(\bar{\boldsymbol{S}}_x \left(\nabla_q \boldsymbol{x}\right)\right)^{\#}\right)^{\#} \boldsymbol{S}_\lambda \left(\boldsymbol{K}_{\lambda,\text{FF}} \dot{\boldsymbol{\lambda}}_{d1} + \boldsymbol{K}_\lambda \boldsymbol{\Delta\lambda}\right) \tag{5.31}$$

The matrix $\boldsymbol{K}_{\lambda,\text{FF}}$ is chosen according to the deformation compensation provided by the trajectory generator.

Due to the varying contact state, we cannot use (5.31) directly. For the swing leg all contacts are inactive, $\nabla_q \boldsymbol{\lambda}$ becomes singular and we cannot compute the pseudoinverse in (5.31). Since the contact force is independent of the trajectory modification $\boldsymbol{\Delta\dot{x}}_\lambda$, we blend from force to position control for components with vanishing contact stiffness:

$$\begin{aligned} \boldsymbol{\Delta\dot{x}}_\lambda = &\left\{\left[\boldsymbol{S}_\lambda \left(\nabla_q \boldsymbol{\lambda}\right) \left(\bar{\boldsymbol{S}}_x \left(\nabla_q \boldsymbol{x}\right)\right)^{\#}\right]^{\#} \boldsymbol{\alpha}_\lambda \boldsymbol{S}_\lambda \left[\boldsymbol{K}_{\lambda,\text{FF}} \dot{\boldsymbol{\lambda}}_{d1} + \boldsymbol{K}_\lambda (\boldsymbol{\lambda}_d - \boldsymbol{\lambda})\right]\right\} \\ &+ \boldsymbol{\alpha}_x \left[\bar{\boldsymbol{S}}_x \boldsymbol{K}_{\lambda x} (\boldsymbol{x}_d - \boldsymbol{x})\right] \end{aligned} \tag{5.32}$$

Here $\boldsymbol{\alpha}_\lambda + \boldsymbol{\alpha}_x = \boldsymbol{E}$ are gain matrices with elements $\alpha_{i,jk} \in [0, 1]$ that determine which dimensions are position or force controlled. Additionally, a damping term is added to each vanishing row of $\left(\nabla_q \boldsymbol{\lambda}\right)$ to enable calculation of the pseudoinverse. Since the corresponding elements in $\boldsymbol{\alpha}_\lambda$ are zero, the magnitude of the damping term does not influence the result of (5.32). Finally, the modified task-space trajectories $\boldsymbol{x}_{d1}$ are calculated as:

$$\begin{aligned} \dot{\boldsymbol{x}}_{d1} &= \dot{\boldsymbol{x}}_{d0} + \bar{\boldsymbol{S}}_x^T \boldsymbol{\Delta\dot{x}}_\lambda \\ \boldsymbol{x}_{d1} &= \boldsymbol{x}_{d0} + \bar{\boldsymbol{S}}_x^T \boldsymbol{\Delta x}_\lambda \\ \boldsymbol{\Delta x}_\lambda &= \int \boldsymbol{\Delta\dot{x}}_\lambda dt \end{aligned} \tag{5.33}$$

For safety reasons, $\boldsymbol{\Delta x}_\lambda$ and $\boldsymbol{\Delta\dot{x}}_\lambda$ are limited to reasonable values. During normal walking, these limits are not reached. The method of calculating the modified task-space velocities $\dot{\boldsymbol{x}}_{d1}$ is illustrated in Figure 5.4.

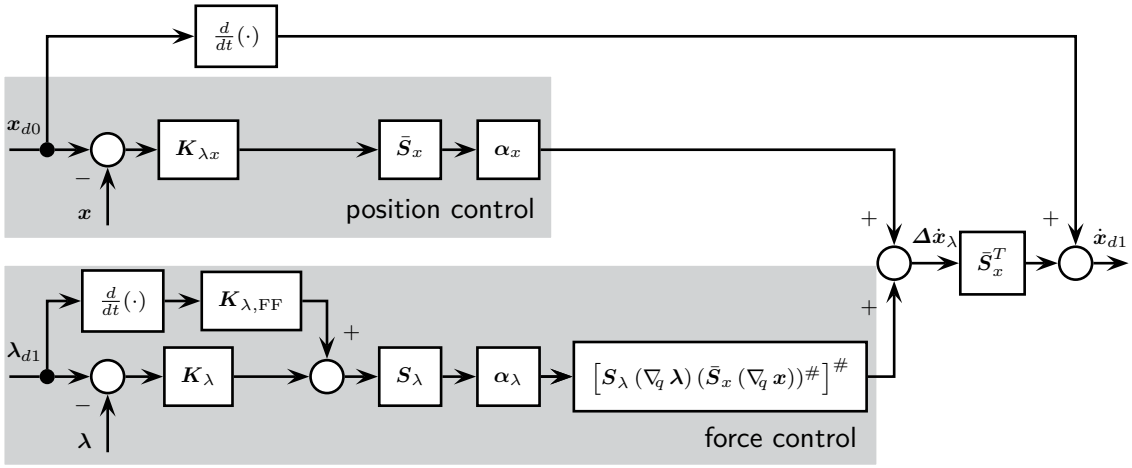


Figure 5.4: Proposed method of modifying task-space trajectories for hybrid position/force control

5.4 Inverse Kinematics and Redundancy Resolution

In the last processing stage before joint position control, joint angles that track the modified reference trajectories $\boldsymbol{x}_{d1}$ are determined. That is, a solution to the equation

$$\boldsymbol{x}(\boldsymbol{q}) = \boldsymbol{x}_{d1} \tag{5.34}$$

is calculated. In the following, $\boldsymbol{x}$ is a reduced vector of the full set of task-space coordinates listed in Section 4.4.2 and Table 4.2. For Lola, the pelvis joint angles are removed from $\boldsymbol{x}$ in the default implementation. Then, $\dim(\boldsymbol{x}) < \dim(\boldsymbol{q})$, the robot is kinematically redundant and there is, in general, no unique solution. This offers the opportunity for optimization, since we can choose trajectories that not only track $\boldsymbol{x}_{d1}$, but also minimize a cost function H.

Minimizing H for the entire trajectory is not possible in real-time, since it involves solving a complex optimal control problem [71]. Therefore, a "local optimization" [71] of kinematic redundancy is performed at the velocity level. Differentiating (5.34) leads to a linear equation for $\dot{\boldsymbol{q}}$:

$$\left(\nabla_q \boldsymbol{x}\right)\dot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{x}}\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}_{d1} \tag{5.35}$$

The well-established method of *resolved motion rate control* originally proposed by Whitney [130] is chosen to calculate $\dot{\boldsymbol{q}}$. The idea is to calculate $\dot{\boldsymbol{q}}$ as the solution of a constrained quadratic programming problem:

$$\begin{aligned} &\frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{W}\dot{\boldsymbol{q}} \rightarrow \min! \\ &\dot{\boldsymbol{x}} - \boldsymbol{J}_{\boldsymbol{x}}\dot{\boldsymbol{q}} = \boldsymbol{0} \end{aligned} \tag{5.36}$$

The solution to this problem minimizes the required joint velocities for tracking $\dot{\boldsymbol{x}}$, where the individual joint speeds are weighted by the (usually diagonal) matrix $\boldsymbol{W}$.

Liégeois [58] proposed minimizing a cost function $H(\boldsymbol{q})$ by modifying (5.36). The change in H during one control cycle is:

$$\Delta H = (\nabla_q H)\, \Delta \boldsymbol{q} \approx (\nabla_q H)\, \dot{\boldsymbol{q}} \Delta t \tag{5.37}$$

Therefore, the largest reduction of H during one control cycle is given by $\dot{\boldsymbol{q}} = -(\nabla_q H)^T$. Instead of choosing this direction of "steepest descent" for H, a modified cost function for minimizing both H and the required joint speeds is chosen:

$$\begin{aligned} \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}} - \alpha_N (\nabla_q H)\, \dot{\boldsymbol{q}} &\rightarrow \text{min!} \\ \dot{\boldsymbol{x}} - \boldsymbol{J}_{\boldsymbol{x}} \dot{\boldsymbol{q}} &= \boldsymbol{0} \end{aligned} \tag{5.38}$$

The scalar parameter $\alpha_N > 0$ determines the balance of priorities between minimizing H and minimizing $\frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}}$. The solution to (5.38) can be computed using the method of Lagrange multipliers. The closed form solution is:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}_{\boldsymbol{x},\boldsymbol{W}} \dot{\boldsymbol{x}} + \underbrace{\left(\boldsymbol{E} - \boldsymbol{J}^{\#}_{\boldsymbol{x},\boldsymbol{W}} \boldsymbol{J}_{\boldsymbol{x}}\right)}_{\boldsymbol{N}} \boldsymbol{W}^{-1} \alpha_N \nabla_q H^T \tag{5.39}$$

Here $\boldsymbol{J}^{\#}_{\boldsymbol{x},\boldsymbol{W}}$ is the $\boldsymbol{W}$-weighted generalized inverse and $\boldsymbol{N}$ a null-space projection matrix. While this closed form solution is often used, it is significantly more efficient to directly compute $\dot{\boldsymbol{q}}$ from the optimization problem without calculating $\boldsymbol{J}^{\#}_{\boldsymbol{x}}$ and $\boldsymbol{N}$. This approach, proposed by Klein and Huang (see [71]), is detailed in Appendix F.

For Lola, the objective function H is used to avoid joint limits and to choose symmetric and "comfortable" poses. Quadratic cost functions are chosen for both components. However, joint limit avoidance is only activated close to the edges of the working range. The resulting cost function is shown schematically in Figure 5.5.

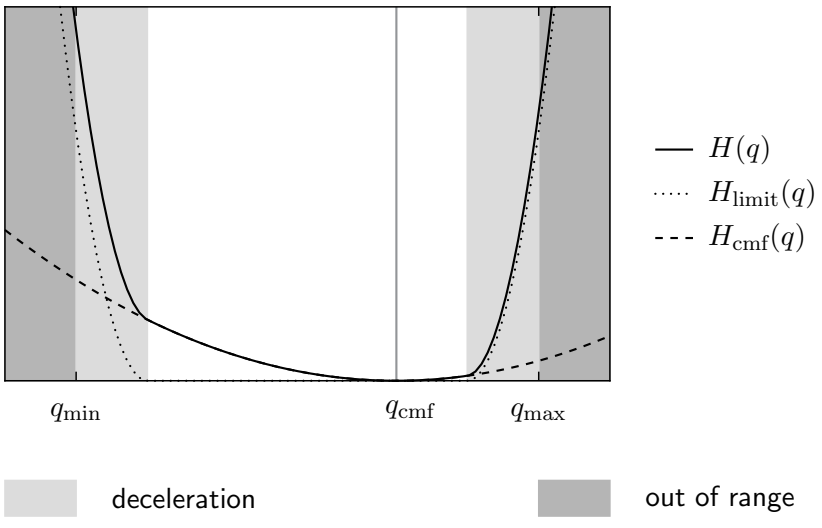


Figure 5.5: Schematic representation of the objective function $H = H_{\text{limit}} + H_{\text{cmf}}$ for a single joint. $H_{\text{limit}}, H_{\text{cmf}}$ are the terms for joint limit avoidance and convergence towards the preferred joint angle q_{cmf}, respectively.

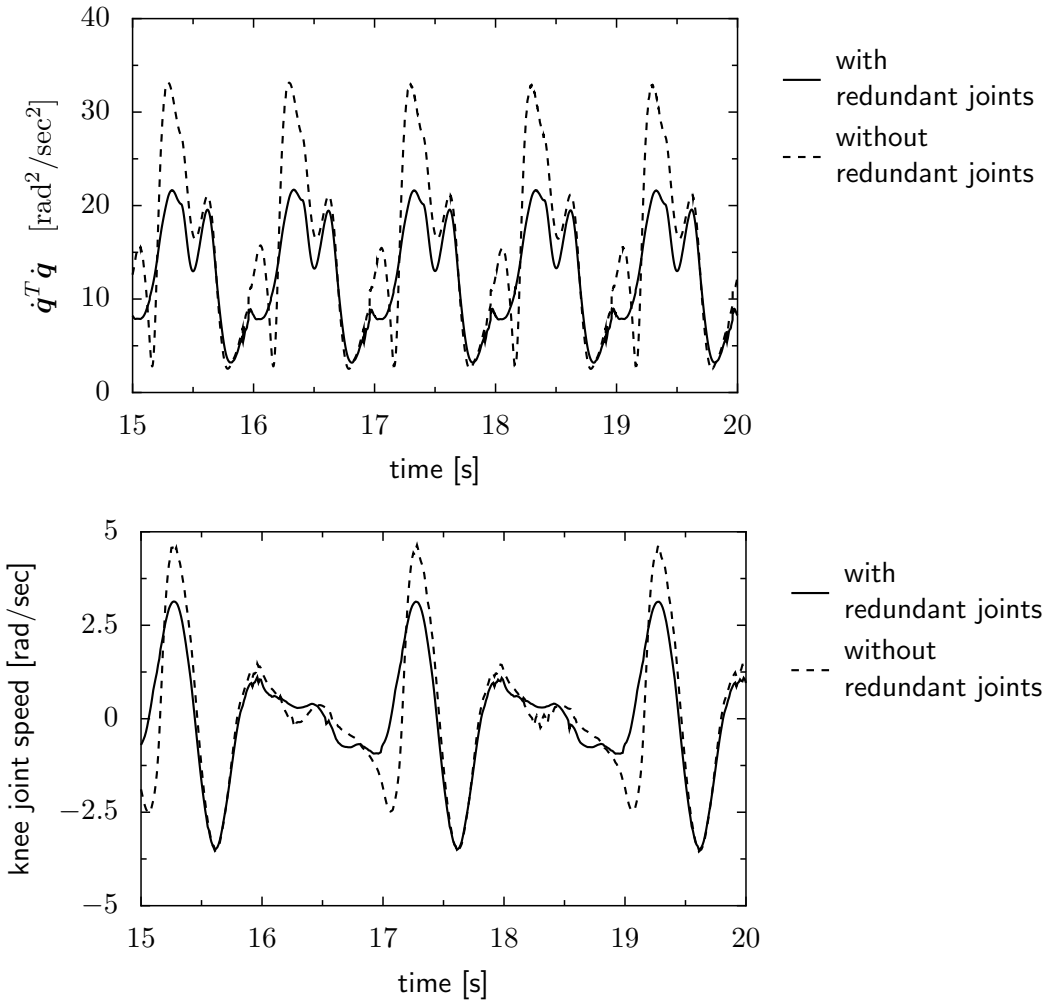


Figure 5.6: The required angular velocities can be decreased by using redundant toe and pelvis joints: both the square of generalized velocities $\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}}$ (top) and the knee joint speeds (bottom) are significantly reduced.

Figure 5.6 illustrates the effect of the redundant toe and pelvis joints on the required joint speeds. The improvement is the most pronounced for fast moving joints such as the knee.

The desired generalized coordinates are calculated by numerically integrating (5.39). Since the integration is not exact, simply integrating $\dot{\boldsymbol{q}}_{d1}$ would lead to an unbounded error in tracking the task-space trajectories $\boldsymbol{x}_{d1}$. To avoid this, a position control term $\boldsymbol{K}_x(\boldsymbol{x}_{d1} - \boldsymbol{x})$ is added to $\dot{\boldsymbol{x}}_{d1}$. Figure 5.7 illustrates the resulting inverse kinematics algorithm with null-space optimization.

Figure 5.8 illustrates the structure of the entire stabilizing control system. Either the measured generalized coordinates $\boldsymbol{q}$ (feedback path 2) or the desired values from the last control cycle $\boldsymbol{q}_{d1}$ (feedback path 1) can be used as input to the direct kinematics. When feedback path (1) is selected, the term $\boldsymbol{K}_x(\boldsymbol{x}_{d1} - \boldsymbol{x})$ compensates the numerical drift of time integration. For feedback path (2) this term constitutes a task-space position control. Experiments with both versions showed similar performance. In most cases, the numerical drift compensation is chosen, since

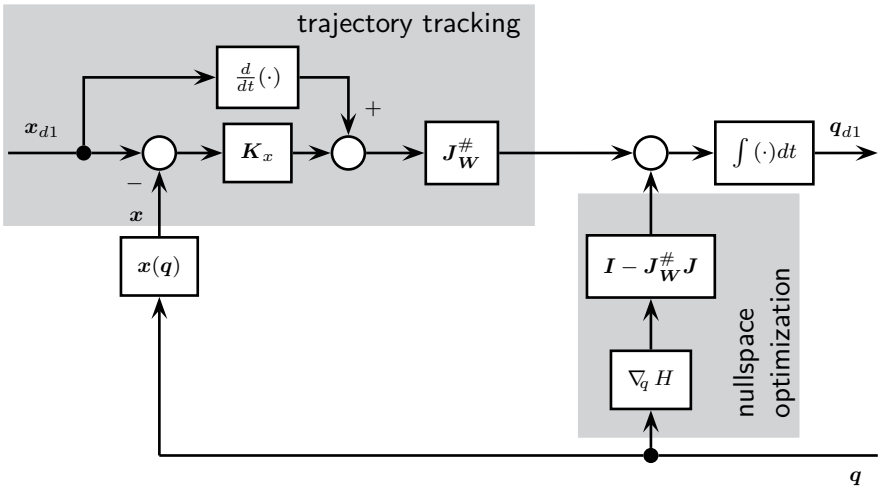


Figure 5.7: Inverse Kinematics with null-space optimization

it proved to be more robust.

A different method for local optimization of redundancy was previously proposed by the author for controlling the robot H7 [8]. In this approach, a base of the null-space is explicitly constructed using a singular value decomposition (SVD) of the Jacobian matrix. This approach provides several advantages and was successfully used for controlling a kneeling motion of H7, as shown in Figure 5.9. Having an explicit representation of the null-space enables more fine-grained control over the local optimization procedure. However, the numerical cost of calculating the SVD proved too high for real-time use in the walking controller, due to the larger number of task-space degrees of freedom.

5.5 Joint Position Control

5.5.1 Joint Position Control for Johnnie

The lowest level in the walking control system is a joint position control loop. Johnnie does not have current sensors, so the position is directly controlled by modifying the armature voltage $\boldsymbol{U}$ depending on the measured ($\boldsymbol{\varphi}_{\text{mot}}$) and desired ($\boldsymbol{\varphi}_{\text{mot},d}$) angles of the motor shafts:

$$\boldsymbol{U} = \boldsymbol{C}(\boldsymbol{\varphi}_{\text{mot},d} - \boldsymbol{\varphi}_{\text{mot}}) + \boldsymbol{D}(\dot{\boldsymbol{\varphi}}_{\text{mot},d} - \dot{\boldsymbol{\varphi}}_{\text{mot}}) + \boldsymbol{k}_M \dot{\varphi}_{\text{mot},d} \tag{5.40}$$

The gain matrices $\boldsymbol{C}, \boldsymbol{D}$ are diagonal, i.e., the joint position control is decoupled. Nevertheless, this control loop is executed on the on-board PC with the same sampling time of 1.5 ms used for all controllers on Johnnie. Since the motor side of the joint is controlled, the ankle joint kinematics do not have to be considered in this control loop.

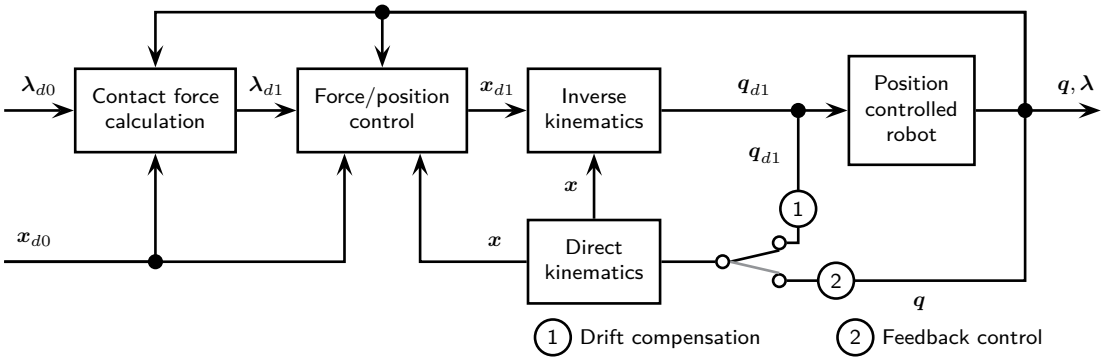


Figure 5.8: Schematic overview of the stabilizing control system. When feedback path (1) is selected, the system uses a pure inverse kinematics algorithm. Choosing feedback path (2) leads to a task space position control.

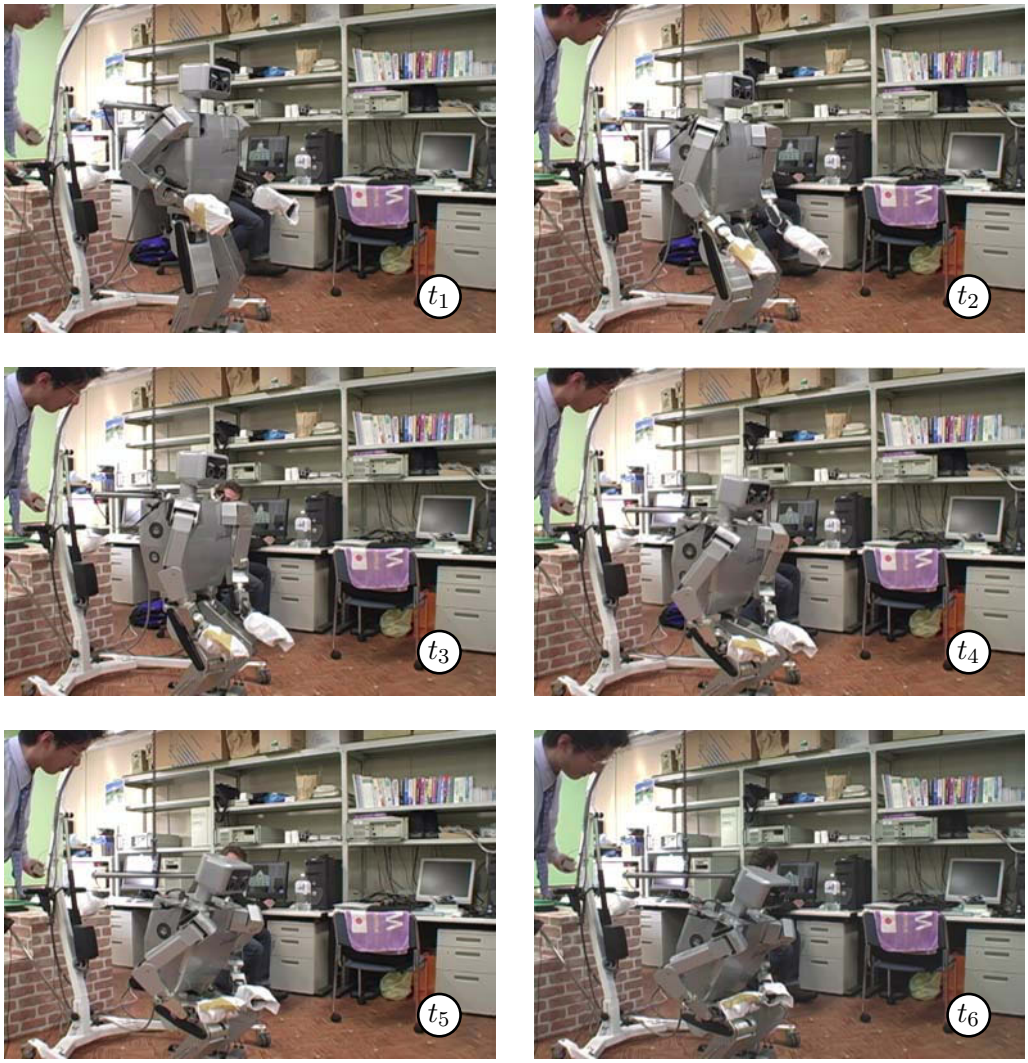
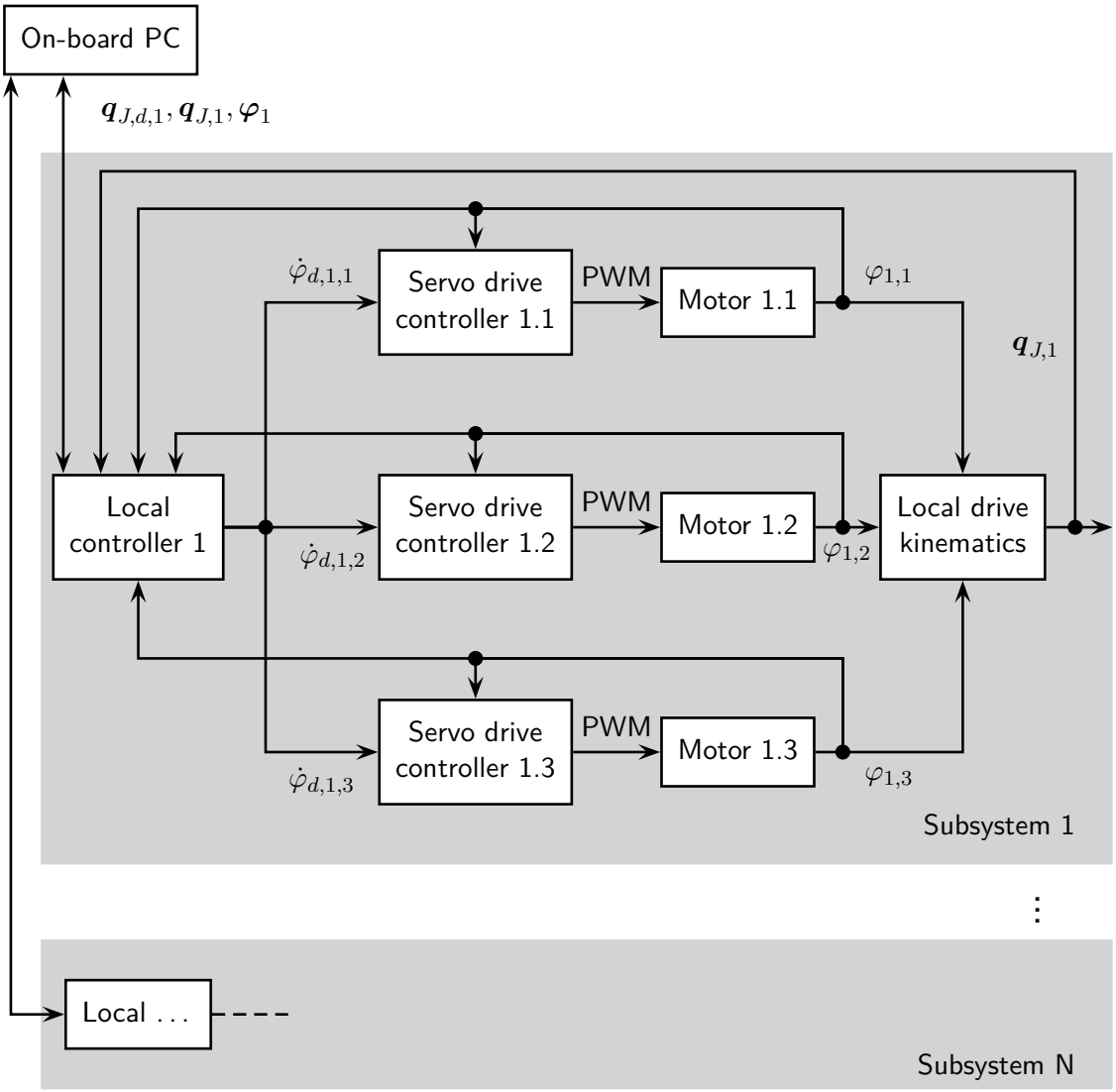


Figure 5.9: A different approach to redundancy resolution was applied to H7 for generating a kneeling motion. The rotation of the hand about the surface normal of the thigh is a result of null-space motion and joint limit avoidance.



$\boldsymbol{q}_{J,d,i}$	Desired joint angles for i-th subsystem
$\boldsymbol{q}_{J,i}$	Measured joint angles for i-th subsystem
$\varphi_{d,i,j}$	Desired motor shaft angles for j-th motor in i-th subsystem
$\varphi_{i,j}$	Measured motor shaft angles for j-th motor in i-th subsystem
PWM	Pulse width modulated input to motor windings

Figure 5.10: Schematic representation of Lola's distributed joint control system

5.5.2 Joint Position Control for Lola

Lola's low-level control is a distributed system with nine local controllers (custom made PCBs), each responsible for two to three joints. The local controllers receive joint angle targets from the on-board PC. Each local controller is connected to either two or three commercial digital servo drive controllers[1], each in turn connected to one motor. Depending on the feedback loop used (see below), the local controllers must calculate forward kinematics and/or Jacobians of the local drive kinematics. For the controllers responsible for knee and ankle joints, the kinematics are quite involved

1 Whistle and Guitar modules from Elmo Motion Control Ltd. (http://www.elmomc.com/)

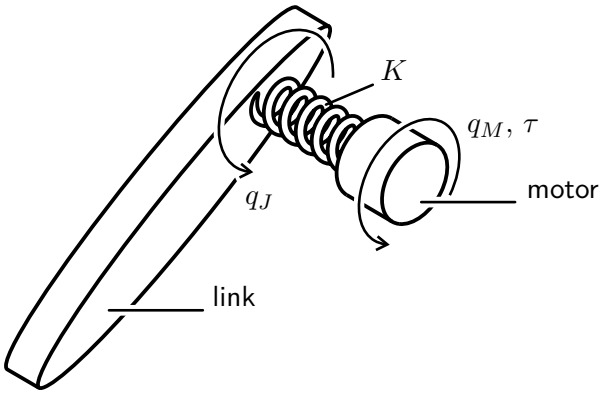


Figure 5.11: Single elastic joint model used for stability analysis

(cf. Section 2.2.3). The controllers do not have floating point units, leading to large rounding errors in certain calculations. Therefore, look-up tables are computed off-line using 64-bit floating point arithmetic and converted into a scaled 32-bit integer representation. Jacobians and motor angles are then calculated by linear interpolation. Figure 5.10 illustrates the structure of the distributed control system. The electronics components were developed by Georg Mayr and Mathias Bachmayer and the local controllers were programmed by Valerio Favot.

Lola is equipped with link and motor-side position sensing. The motor-side incremental encoder signal is connected to both servo drive controllers and the local controller, while the link-side absolute encoder is only connected to the local controller. The servo drive controllers are capable of current and motor-side speed control. This structure enables a number of different control architectures.

While the stiffness of Lola's joints is very high (cf. Section 2.4.3), the elasticity must be taken into account for link-side position control. In the following, the stability properties of several combinations of link and motor-side feedback are analyzed using a simple model of a single elastic joint. For the sake of simplicity, only PD-type control laws are considered. In the implementation, the PD-type control can also be implemented as a cascade control with inner velocity and outer position control. The stability analysis for various cascade control schemes with and without integral control were performed in the same manner.

Single Elastic Joint Model

Basic stability properties of different feedback configurations will be studied using the two mass system shown in Figure 5.11. A similar analysis for the control of elastic joints is given in [110, Ch. 13]. To simplify the analysis and because it is the worst case, damping will be neglected. Furthermore, and without loss of generality, the gear ratio is set to one. With reference to Figure 5.11, the EoM can be written as:

$$J_J \ddot{q}_J + K(q_J - q_M) = 0 \tag{5.41}$$

$$J_M \ddot{q}_M + K(q_M - q_J) = \tau \tag{5.42}$$

Here q_J is the joint angle and q_M is the motor angle.

Link-Side Position and Velocity Feedback

In one possible feedback configuration, only link-side sensing is used, leading to the following control law:

$$\tau = K_p(q_{J,d} - q_J) + K_d(\dot{q}_{J,d} - \dot{q}_J) \tag{5.43}$$

For this arrangement, the transfer function is computed as:

$$G_{PD,\text{link}} = \frac{K(K_p + sK_d)}{(J_M J_J)s^4 + K(J_M + J_J)s^2 + KK_d s + KK_p} \tag{5.44}$$

Stability requires that all poles of the system lie in the left half-plane. According to the Hurwitz criterion [37], link-side feedback is *always unstable*, since the coefficient for s^3 is zero. Experiments with Lola showed that joints with Harmonic Drives and low effective inertia J_J, such as the hip joints, are stable even for link-side PD control because of the good damping characteristics of Harmonic Drive gears. However, joints with high effective inertia and low friction, such as the knee joints, became unstable even for low feedback gains. Interestingly, all joints could be controlled effectively with the robot hanging in the air. When the robot is standing on the ground, however, the effective inertia acting on the knee and ankle joints is much higher, leading to an unstable system.

Motor Side Position and Velocity Feedback

Instability of link-side position control is due to the non-collocation of position measurement and actuator force. Therefore, a simple solution to this problem is to use only motor-side sensing:

$$\tau = K_p(q_{J,d} - q_M) + K_d(\dot{q}_{J,d} - \dot{q}_M) \tag{5.45}$$

Note that $\dot{q}_{J,d} = \dot{q}_{M,d}$, since a gear ratio of one is assumed. Calculating the transfer function yields:

$$G_{PD,\text{motor}} = \frac{K_d K s + KK_p}{(J_M J_J)s^4 + (K_d J_J)s^3 + (J_M K + K_p J_J + KJ_J)s^2 + KK_d s + KK_p} \tag{5.46}$$

For this configuration all coefficients are present and positive. According to the Hurwitz criterion, all poles are in the left half plane if the following inequalities hold:

$$\begin{vmatrix} KK_d & K_d J_J \\ KK_p & J_M K + K_p J_J + KJ_J \end{vmatrix} = K^2 K_d (J_J + J_M) \qquad > 0 \tag{5.47}$$

$$\begin{vmatrix} KK_d & K_d J_J & 0 \\ KK_p & KJ_M + K_p J_J + J_J K & J_J J_M \\ 0 & KK_d & K_d J_J \end{vmatrix} = K^2 K_d^2 J_J^2 \qquad > 0 \tag{5.48}$$

That is, the system is stable for all positive gains when only motor-side feedback is used. These positive characteristics are preserved in the full system, leading to very robust performance. Since very high gains can be used, the motor-side tracking error is small. However, the (small) deformations in the drive cannot be compensated using this feedback arrangement. Note that for this arrangement tracking motor-side errors tend to decrease with increasing gear compliance, while the link-side error increases.

Link-Side Position and Motor-Side Velocity Feedback

One possibility of stabilizing the system while maintaining link-side position feedback is to use the motor-side velocity signals. In this configuration, the control torque is given by:

$$\tau = K_p(q_{J,d} - q_J) + K_d(\dot{q}_{J,d} - \dot{q}_M) \tag{5.49}$$

This leads to the following transfer function:

$$G_{PD,\text{link,mot}} = \frac{(KK_d)s + KK_p}{(J_J J_M)s^4 + (K_d J_J)s^3 + K(J_M + J_J)s^2 + (KK_d)s + KK_p} \tag{5.50}$$

Again, we can check for stability using the Hurwitz determinants:

$$\begin{vmatrix} KK_d & K_d J_J \\ KK_p & K(J_M + J_J) \end{vmatrix} = KK_d[K(J_M + J_J) - K_p J_J] \qquad > 0 \tag{5.51}$$

$$\begin{vmatrix} KK_d & K_d J_J & 0 \\ KK_p & K(J_M + J_J) & J_J J_M \\ 0 & KK_d & K_d J_J \end{vmatrix} = K J_J^2 K_d^2 (K - K_p) \qquad > 0 \tag{5.52}$$

Both inequalities hold for $K_p < K$ and $K_d > 0$, i.e., stable gains are limited by the joint stiffness K. On the physical robot, stability margins are increased by viscous gear friction, but the maximum K_d is limited by finite sampling times and the fact that derivative feedback amplifies high frequency noise. This feedback arrangement worked well in walking experiments with Lola. However, it proved to be somewhat more sensitive to disturbances than pure motor-side feedback.

5.6 Optimization-Based Parameter Tuning

Proposed Method

The walking control system has a large number of parameters that must be set correctly to achieve robust and fast walking. Because of the complex, nonlinear and non-smooth dynamics of the robot, properly setting control gains is difficult. Moreover, there are no well-established methods for choosing many of the parameters. Examples for such parameters include the step length for a given walking speed, the length of the double support phase or the foot mass m_l in the reduced model (4.38) used for trajectory generation.

Therefore, initial guesses for such parameters are determined off-line by numerical

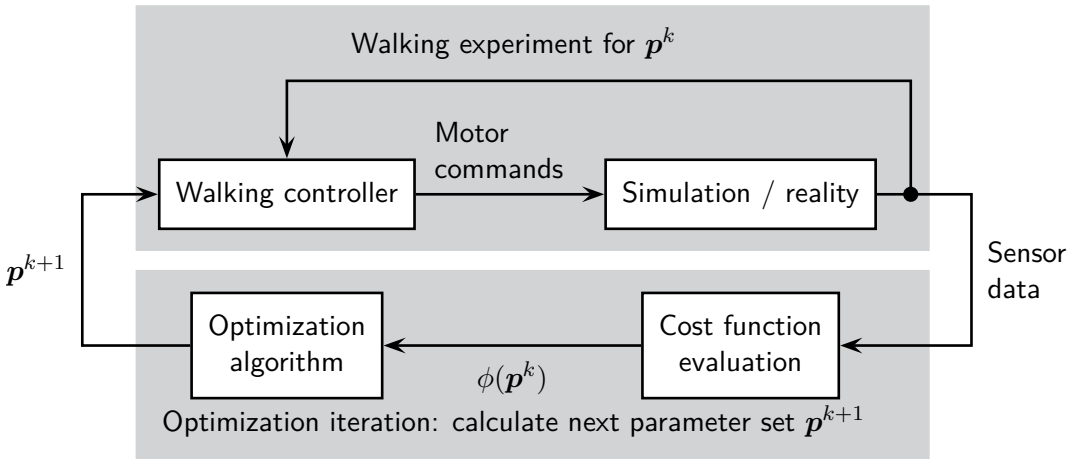


Figure 5.12: Parameters of the trajectory generator and stabilizing controller can be tuned using an optimization algorithm.

optimization using the comprehensive dynamics simulation. This enables a mathematically optimal choice of gait and control parameters for a given objective function. Since every model has a finite accuracy, the optimized parameters are not always optimal for the physical robot. Therefore, the final parameters are determined by manual tuning in walking experiments.

Figure 5.6 illustrates the method of off-line parameter optimization. The optimization algorithm searches for an optimal vector of gait and control parameters $\boldsymbol{p}^\star$ by repeatedly evaluating the cost function $\phi(\boldsymbol{p})$.

Formally, the parameter optimization problem can be written as:

$$\begin{aligned} &\phi(\boldsymbol{p}) = \int f(\boldsymbol{q})dt \rightarrow \min! \\ &\boldsymbol{p}_{\min} \leq \boldsymbol{p} \leq \boldsymbol{p}_{\max} \end{aligned} \tag{5.53}$$

Since calculating $\phi(\boldsymbol{q})$ requires simulating the robot's forward dynamics and controller for several seconds, function evaluations are quite costly. Numerical analysis has shown that the global behavior of many typical cost functions is convex but noisy due to effects like friction and sensor noise. Therefore Implicit Filtering is used as an optimization algorithm, since this method has been successfully applied to many real world problems with noisy cost functions, where function evaluations require expensive numerical simulations [20, 26]. Parallel and serial implementations of the algorithm are available on-line [52].

The implemented system is not only capable of performing off-line optimizations based on dynamics simulations (software in the loop optimization), it is also possible to automatically execute experiments using the actual robot to evaluate the cost function (hardware in the loop optimization). This approach was used for initial tuning of the joint position control loop. However, for a larger number of parameters the required number of function evaluations makes this approach impractical, since, contrary to simulations, experiments cannot be parallelized and must be supervised.

Application Example

An application example for Johnnie is given in the following. The step length L_x, for a given walking speed, the swing leg mass m_l of the simplified model and the height z_{CoG} of the center of gravity were chosen as free parameters. In this example, the following cost function was evaluated for $T = 10\,\mathrm{s}$ after the robot had accelerated to a periodic gait at 2.5 km/h:

$$\phi = \int_{t_0}^{t_0+T} (\Delta\varphi_x^2 + \Delta\varphi_y^2)dt \tag{5.54}$$

Here $\Delta\varphi_x, \Delta\varphi_y$ are the robot's upper body inclination errors in the sagittal and frontal plane respectively. For this example, optimal gait parameters are $L = 0.457\,\mathrm{m}$, $m_l/m = 0.082$ and $H = 0.79\,\mathrm{m}$. An optimization of only L_{step} and z_{CoG} with $m_l = 0$ leads to an optimal cost function value of 5.70510^{-05}, while additionally optimizing m_l leads to a 35% decrease to 3.72810^{-05}. The corresponding decrease in upper body oscillations is shown in Figure 4.17.

5.7 Chapter Summary

In this chapter we presented a model-based approach to stabilizing biped robots by feedback control. Standard control methods for industrial robots are not applicable, since the system is underactuated and there are only unilateral contacts with the environment.

The basic approach is based on an analysis of structural properties of the system dynamics and the observation that overall linear and angular momentum are only influenced by contact forces. Therefore, stabilization of global system dynamics is based on controlling contact forces.

At the next level of the control hierarchy, contact forces and task space trajectories are tracked by a position-based hybrid position/force control that uses an explicit contact model.

Instantaneous joint angle targets for the redundant robot are generated using resolved motion rate control with null-space optimization. This approach makes it relatively easy to utilize redundant DoFs to avoid joint angle limits and reduce angular velocities during walking.

Finally, we gave an overview of Lola's and Johnnie's joint position control with a focus on Lola's distributed control system and the stability properties of different feedback arrangements with link and motor-side position sensing.

6 Autonomous Walking

Lola was developed as part of the DFG proposal "Ein leistungsgesteigerter, autonomer Zweibeiner" (An Autonomous Biped with Enhanced Performance). One goal of this research project is to develop a biped that can *autonomously* navigate in unknown environments.

A truly autonomous robot does not require any external supply of energy, sensor data or computing power and does not need to be supported or supervised by humans. While integrating batteries and additional computers into the robot is an engineering challenge, it is one that can be solved to a certain extent using current off-the-shelf technology. Moreover, research on improved battery technology for electric and hybrid vehicles and the continuing improvement of microprocessors will solve these problems of "physical" autonomy in the foreseeable future. Therefore, the focus in this research project was put on developing methods required for navigating in unknown environments, using only on-board cameras, while the power supply and vision processing computers are located outside of the robot. The computer vision system itself is being developed by the Institute for Autonomous Systems Technology at University of the Bundeswehr in Munich, Germany.

This chapter gives a brief overview of the computer vision system and describes the integration of walking controller and computer vision.

6.1 Computer Vision System

This thesis does not cover computer vision. Nevertheless, a brief overview of the system developed for Lola is given in order to provide the necessary context. A more detailed overview of the approach to autonomous locomotion is given in [10].

The on-board camera system consist of two high-resolution five megapixel CCD cameras with a maximum of 17 frames per second (Basler pilot piA2400-17gm). Because of the very large amount of data, vision processing is performed on three external computers with dual Intel Xeon Quad-Core CPUs (central processing units). Camera data is sent to the external computers via two Gigabit Ethernet connections.

Most previous work on autonomous locomotion has focused on building environment models in which a planner can search for feasible paths that are then sent to the robot controller [29, 42, 82, 105]. This requires a complex and computationally expensive technique for simultaneous localization and mapping (SLAM). Following a very different philosophy, Lola's vision system instead classifies a set of pre-calculated paths or "visual tentacles" as passable or not. Solving this classification problem does not require building an explicit environment map or prior object models. Furthermore, since no obstacle templates or color coding are used, the approach is quite general and works for arbitrary, previously unknown objects. The approach was

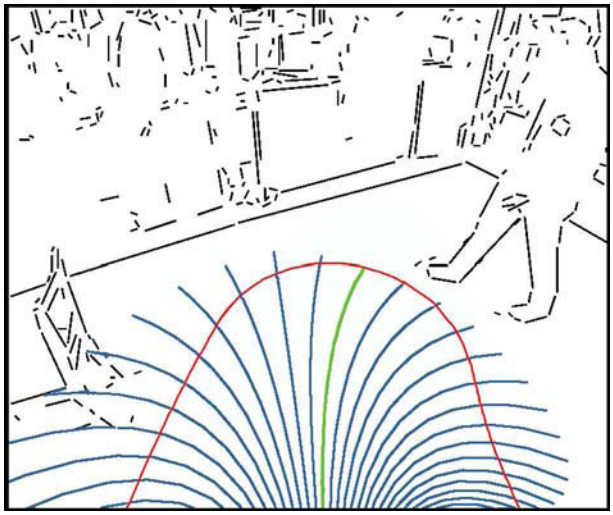


Figure 6.1: Example for tentacle based navigation: detected contour segment features are shown in black, projected tentacles in blue and the actually selected tentacle in green, surrounded by the corresponding classification area shown in red (image taken from [10]).

originally developed by VON HUNDELSHAUSEN et al. for autonomous vehicles [127] and then adapted for humanoid robots.

The tentacles are circular paths with different curvatures represented in a robot-centric coordinate system. This choice of tentacles integrates very well with the step sequence planner of Lola's real-time controller, which also uses piecewise circular paths (cf. Section 4.3).

The target path is chosen by rating all tentacles using camera image data. At the time of writing (August 2010), the lowest level of Lola's vision system has been completed. At this level, the robot tries to avoid all obstacles. The rationale is to provide a robust and general obstacle avoidance mechanism that allows the robot to safely operate in unknown environments. More complex behavior involving specific actions for well-defined objects such as stairs or tables will be added on top of this basic layer. Therefore, the distance to the closest obstacle is used for rating potential paths. Tentacles outside the field of view are disabled. To avoid unnecessary changes in the walking direction, tentacles similar to the one chosen during the last sampling period are favored. If no feasible tentacle is found, the robot turns on the spot until a new path is detected.

To determine the distance to potential obstacles, the 3-D tentacles are projected into both camera images. The images are then searched in an area surrounding the tentacle for features indicating obstacles. This is very efficient, since only the part of an image relevant to potential actions is analyzed. Since every object has a contour, finding obstacles is based on extracting potential contours using edge detection. Effectively, the 3-D search for a feasible path is reduced to a 2-D classification problem, completely avoiding SLAM. The selected path is converted into an instantaneous linear and angular velocity and sent to the walking controller.

Figure 6.1 shows contour segment features extracted from a camera image taken at the *Hannover Messe 2010* (cf. Chapter 8) with tentacles projected into the image.

6.2 Interfacing Walking Control and Computer Vision

Both computer vision system and walking control use local coordinate systems centered on the robot. However, since vision and control systems run on different computers and with different cycle times, it is important to establish a common frame of reference. Therefore, the walking controller estimates the position and orientation of its local coordinate system I_i with respect to the global frame I_0 using odometry (cf. Section 4.4.1). The estimate is sent to the vision system together with the current robot pose. Kinematic data and camera images can be precisely synchronized and time-stamped, since the cameras are triggered by the local controller on Lola's head. The physical connections of vision system and walking controller are illustrated in Figure 6.3.

The vision system specifies motion commands in the local, robot-centric frame I_i, while viewing targets are specified in the global frame I_0 (cf. Section 4.7). Note that the estimated global position is only used for communication purposes and does not have to be exact, since both control and vision processing are performed in local coordinate frames.

While the vision system needs some information about the robot's configuration, the detailed kinematic model with all DoFs is not required. Therefore, relevant information is condensed into an abstract robot model with two cameras and telescopic legs, whose configuration is represented by five coordinate systems: one for each foot and each camera and one for the upper body. Figure 6.2 illustrates the kinematic data used by the computer vision system.

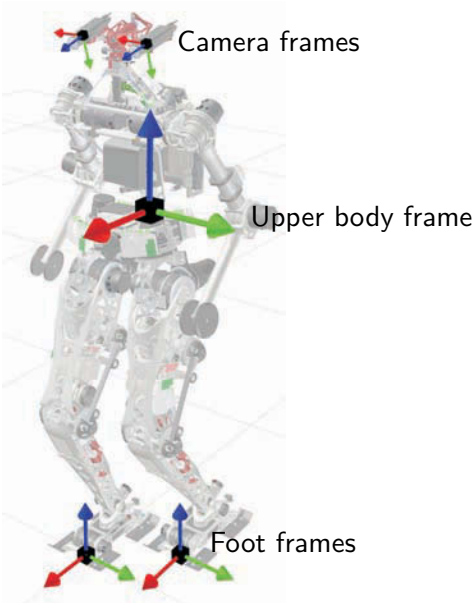


Figure 6.2: Kinematic data precisely synchronized with camera images is sent to the external vision processing system.

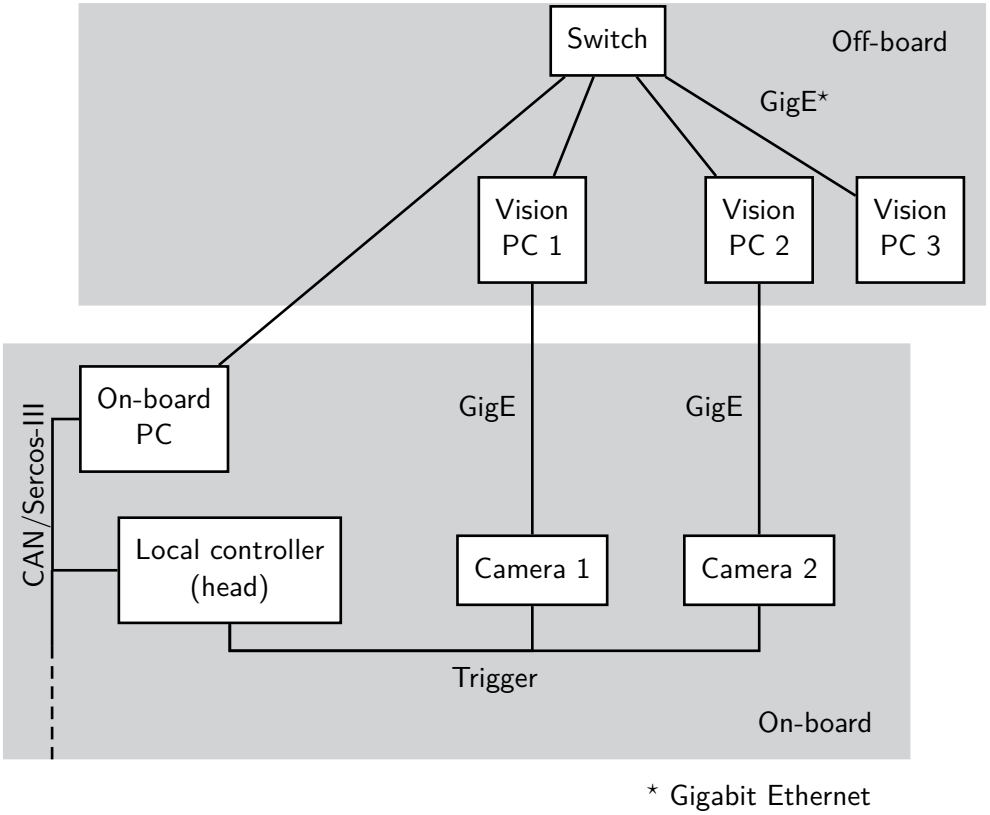


Figure 6.3: Physical connections of walking control and vision processing

7 Software System

7.1 Introduction

Motivation

Simulation and control of such complex mechatronic systems as humanoid robots requires an extensive software system. In fact, "software architecture" is a major topic in current robotics research (IEEEXplore[1] returns more than 1000 results to a query for "software architecture" and "robotics"). This thesis does not try to develop a generic software architecture for intelligent and/or walking robots. However, a versatile and efficient software system is a prerequisite for both simulation-based analysis and design and experimental research.

This chapter gives an overview of the system developed by the author for Johnnie and Lola.

Design Approach

There are a number of conflicting requirements for the various software components:

Efficiency: Both dynamics simulation and real-time control require efficient algorithms and implementations. For simulations, the emphasis is on minimizing the total run-time of the program. For real-time control, the worst case execution time for one control cycle must be minimized.

Portability: Software components are used for both simulation and real-time control. While simulations are run on a general purpose operating system, the real-time control must be executed on a real-time operating system (RTOS). Therefore, use of platform-specific application programming interfaces (API) and libraries must be avoided whenever possible.

Code Reuse: To minimize the amount of code, software components should not only be reused on different operating systems (OS), but also shared among different applications.

Flexibility: The software system is a tool for research on humanoid walking robots. Therefore, modifying models and controllers should be as simple as possible.

Minimal Development Time: Since software is not the subject but only a necessary tool for this research, development time should be minimized.

Safety: Johnnie and Lola are expensive, powerful and dangerous devices and small errors can easily damage the robot and severely injure the operator. Therefore, safety is essential for all real-time components.

1 http://ieeexplore.ieee.org

All major components of simulation and control programs are written in C++ for maximum efficiency. Smaller utility programs without high demands on efficiency are written in Python in order to speed up development and leverage the large number of existing Python modules.

The approach to portability is to use only standardized, non-proprietary interfaces whenever possible. The Portable Operating System Interface (POSIX) [39] was chosen, since it is supported to some extent by virtually all modern operating systems. Comprehensive support is available on most UNIX-like systems, such as Linux[2] or Solaris[3], and also for RTOSs such as QNX[4] and VxWorks[5]. Microsoft Windows supports only a small subset of the full POSIX standard. Exceptions are made only for the low-level controller, which uses QNX-specific APIs for hardware access, and the joystick interface, which uses the Linux joystick API.

There are two specific measures towards increasing the safety of the real-time system. First, special care is taken to avoid common programming errors by checking return values of all functions, checking for invalid pointers and violated array boundaries. Second, each program implements one clearly defined functionality. This leads to a system of small, cooperating programs instead of one very large program. While this adds a certain amount of complexity and overhead, safety is increased since errors such as invalid pointers in one part of the system cannot spread to other parts. For example, a programming error in a logging application cannot lead to erroneous joint target values being sent to the joint controllers. Instead, only the logging application will crash or work incorrectly. In addition, this decomposition greatly simplifies debugging, since the source of errors can be traced more easily.

7.2 Software Components

To facilitate code reuse, the software is organized into a number of reusable libraries. An overview of the core libraries is given in Figure 7.1. In the following, the functionality of the these libraries is briefly summarized.

Utilities

This library implements basic utility functions used by many other libraries and programs. The functionality includes:

- Efficient logging during real-time execution using circular buffers and multi-threading.
- Utility classes for inter-process communication (IPC) based on the Internet Protocol (IP), message queues, shared memory and a custom, topic based publish/subscribe system.
- If a program encounters an exception or error, utility functions enable it to save the last few seconds of process data and print a stack trace. This greatly simplifies finding both hardware and software faults.

2 http://www.kernel.org
3 http://www.oracle.com/solaris
4 http://www.qnx.com/
5 http://www.windriver.com/products/vxworks/

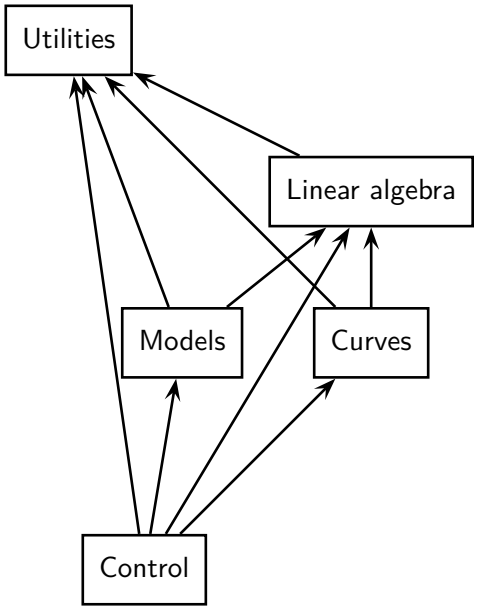


Figure 7.1: Major software libraries developed for Lola and Johnnie

Vector Matrix Library

The vector matrix library is based on a library previously used for other simulation and control applications, such as Johnnie and the eight-legged pipe crawling robot Moritz [89]. Over the years, many people at the Institute of Applied Mechanics have contributed to this library. It provides basic math functions such as addition and subtraction, but also linear algebra functions for solving systems of linear equations or computing the singular value decomposition of a matrix.

Curves

This library provides classes for piecewise fifth-order polynomials, cubic splines, piecewise linear functions and other curves. There also are functions for analytically calculating spline gradients and optimizing parameters of piecewise fifth-order polynomials with respect to required accelerations.

Model Library

The model library provides both generic classes for multibody systems with tree structure and special models for Johnnie and Lola, such as harmonic drive friction and nonlinear drive kinematics. In addition, classes for polygonal environment models and contact solvers are part of this library. The models are implemented using an object oriented approach and run-time polymorphism, i.e., inheritance and virtual methods, in order to maximize code reuse and provide generic interfaces.

Controller Library

The controller library provides both basic components such as digital filters and specific functions for biped walking such as inverse kinematics, step sequence planning and trajectory generation. Core functionalities such as trajectory and step sequence planning, stabilizing control and redundancy resolution only use generic interfaces such as the “Robot” class and can be used without modification for both Johnnie and Lola.

7.3 Main Programs

The software system includes a number of programs for simulation and control, user interfaces and smaller utility programs. This section gives a brief overview of these programs.

Simulation

There are programs for simulating the forward dynamics of Johnnie and Lola using either the full or reduced models (cf. Chapter 2).

Walking Control

The walking controller is separated into walking pattern generator (cf. Chapter 4) and stabilizing controller (cf. Chapter 5). This enables the optimal use of the on-board computer, which is equipped with a dual-core CPU. Joint position control and/or hardware access is performed in a third program. This low-level controller provides a uniform interface to the other programs. This makes higher levels of the controller independent of the robot's electronics architecture, which is very different for Johnnie and Lola.

Autonomous Locomotion

Interfaces to the vision processing system (cf. Chapter 6) are provided by two programs. The first program estimates the robot state using sensor data synchronized with the camera images. The estimated state is then sent to external vision processing computers via UDP (User Datagram Protocol). The second program receives target velocities and fixation points via UDP. These are checked for errors and then forwarded to the walking controller using the publish/subscribe system described in Section 7.4.

User Interfaces

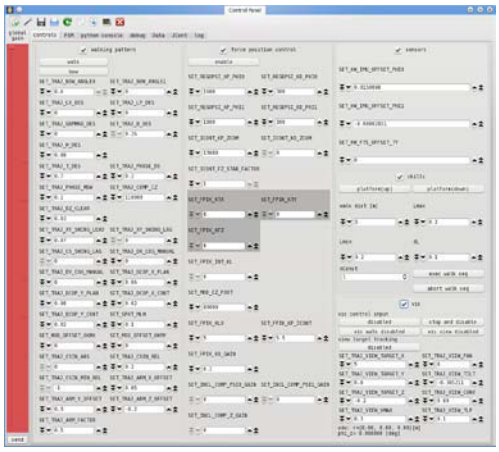
There are two programs for controlling the robot: a graphical user interface (GUI) and a joystick interface. The joystick interface allows the operator to intuitively control the robot by inputting the desired average angular and translational velocity. The GUI provides graphical controls for freqently used gait and control parameters. It also includes an interactive Python console that provides direct access to the publish/subscribe system, allowing the operator to access seldomly used functions. Finally, the GUI provides access to other utility programs such as a graphical plotting application for log files written by different parts of the real-time system. Figure 7.2 shows different views of the GUI.

Analysis and Visualization

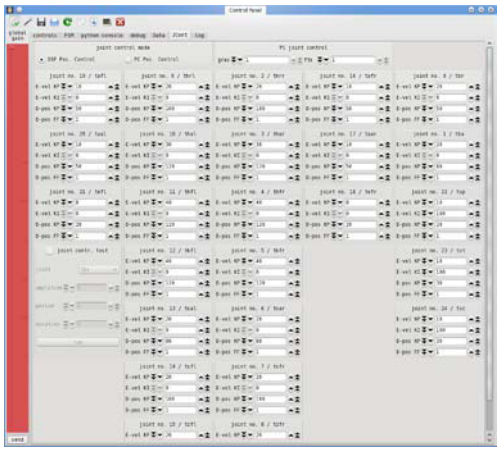
Log files written by the real-time system contain a header detailing the contents of the file. A plotting application parses the header in order to provide a graphical interface useful for quickly analyzing sensor data and internal controller variables.

Animation and visualization of simulation results, data gathered during experiments, as well as a real-time visualization of the robot's state is provided by an OpenGL-based 3-D viewing application[6]. The program was developed in cooperation with Markus Schwienbacher. The application itself provides basic functions for

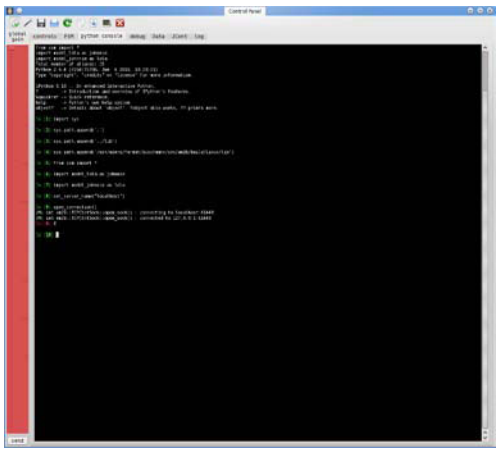
6 OpenGL is a platform independent API for developing 2-D and 3-D graphics applications (`http://www.opengl.org`).



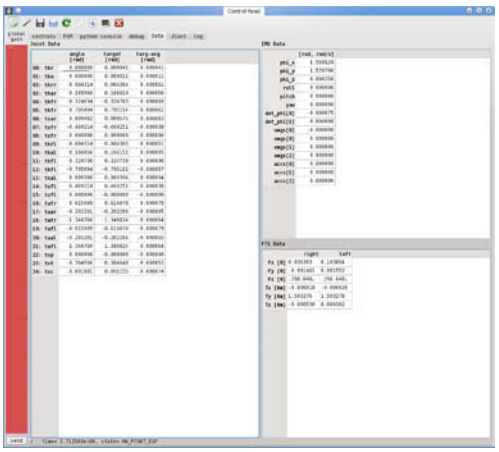
(a) Graphical interface to the main walking control parameters



(b) Graphical interface to the main joint controller parameters



(c) Interactive Python console for accessing all controller parameters



(d) Real-time data display

Figure 7.2: Graphical user interface for Lola and Johnnie

panning, zooming, object selection, exporting still images, movies and static 3-D scenes. Drawing of 3-D objects such as a robot or an obstacle is implemented in plug-ins. This modular approach makes it easy to develop small, problem-specific plug-ins for visualizing simulation or planning results. Currently, the system includes the following plug-ins (among others):

- Animation of Lola
- Animation of Johnnie
- Rendering of static 3-D objects such as obstacles
- Visualization of planned foot, center of gravity and center of pressure trajectories, as well as support polygons
- Animation of contact forces

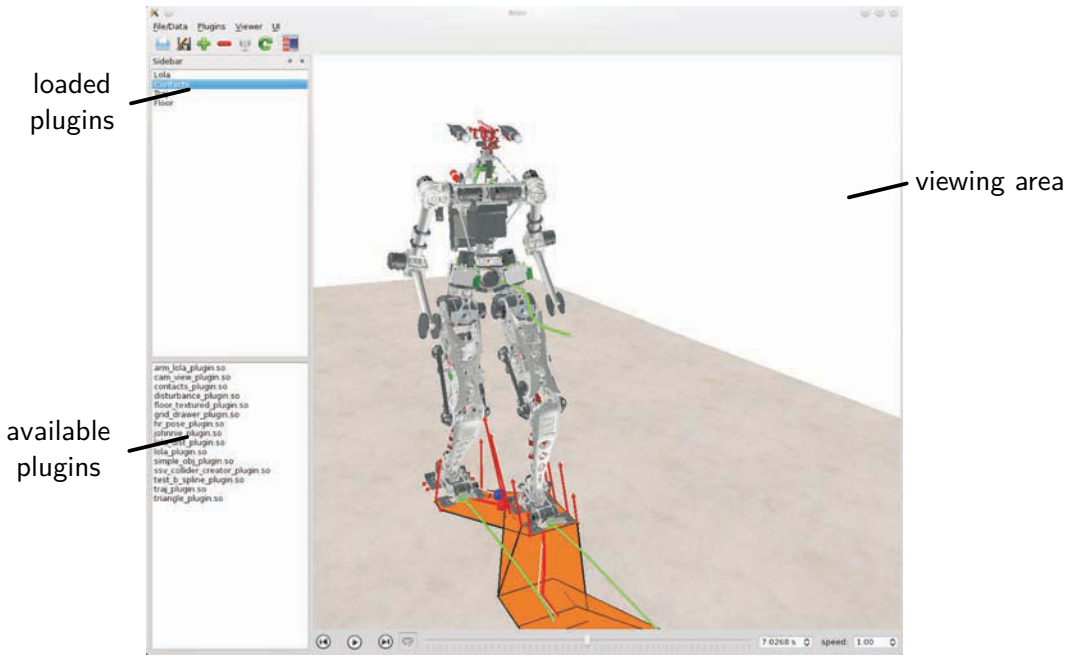


Figure 7.3: 3-D viewing application with a number of plug-ins loaded

Figure 7.3 shows a snapshot of the viewing application with plug-ins for Lola, contact forces and trajectories loaded.

7.4 Real-Time System

The real-time system is organized as a group of cooperating processes. Functions are implemented in different processes rather than threads, since the separation of address spaces enhances the safety and reduces the probability of fatal crashes.

QNX Neutrino is chosen as RTOS, since it provides fully POSIX-compliant interfaces and shows excellent real-time performance, e.g., very fast context switching, synchronization primitives and message passing. Also, as a microkernel OS, it fits well with the design philosophy of the walking control system. Additionally, it provides excellent development and debugging tools.

Inter-process and Intra-process Communication

Since the real-time system is composed of several different processes, it requires some form of IPC. This could be achieved in an ad hoc manner using basic OS services. Such an approach, however, leads to a tight coupling of all components, thereby reducing the flexibility of the system and making changes to it more difficult.

Such concerns are addressed by middleware systems such as CORBA[7] or ICE[8]. In fact, CORBA is used in OpenHRP, the software system used for the HRP robots [46]. However, such middleware systems are heavyweight and require a relatively large amount of supporting code. Moreover, most of the functionality they provide would not be used for Lola and Johnnie.

7 http://www.corba.org
8 http://www.zeroc.com/ice.html

Therefore, a simple publish/subscribe system suitable for hard real-time use was developed. In a publish/subscribe system, the messages are not directly sent from producers to consumers, but instead sent to an intermediary instance, called a message broker. Clients can subscribe to certain types of messages which are then forwarded to them by the broker.

In the following, the basic design of the system implemented for Johnnie and Lola will be presented. Message types are encoded in a message ID that specifies a domain and a sub-ID. Examples of domains are "logging," "trajectory generation" or "joint control." The sub-IDs specify a certain parameter or event, such as "step length," "start walking" or "start logging data." The system is separated into two layers: intra-process and inter-process communication.

Within one process, objects subscribe to specific message IDs. Published messages are directly forwarded by a message broker object (the local broker) using function calls, which is very efficient. The advantage of this design is the loose coupling between the objects constituting a program, which simplifies changes to the controller. The local publish/subscribe system is also used to send events within the finite state machines used to coordinate the walking controller.

Objects may also publish messages to other applications by marking them as global. Global messages are forwarded by the broker object to a message broker application (the global broker) via POSIX message queues. To receive global messages, a process subscribes to an entire domain. Messages of this domain are sent to the local broker via POSIX message queues, from where they are forwarded to objects that have subscribed to the specific message ID. There is one queue per client process and one dedicated, system-wide queue for publishing messages. There is a further, low-priority service queue used for tasks such as registering with the broker and subscribing to new messages. The message broker uses multiple threads to process messages or add new subscriptions. QNX's priority inheritance mechanism ensures that messages sent by high-priority processes are processed first and low-priority messages and service messages are processed later.

Applications can also communicate with the message broker using TCP (Transmission Control Protocol), enabling communication among networked computers. Figure 7.4 illustrates the structure of the publish/subscribe system.

While the publish/subscribe system is very fast, it is not well suited for large amounts of data or data exchanged at a very high frequency. Therefore, shared memory regions are used for sensor data and target joint angles.

System Overview

Figure 7.5 gives an overview of Lola's run-time system. The local controllers are connected to the on-board PC by a field bus. The results presented in this theses were obtained using a CAN bus based system with point-to-point connections for every controller. At the time of writing (August 2010), Lola is being equipped with the fast, Ethernet-based Sercos-III[9] interface, which will replace the CAN-bus connections between PC and local controllers.

The walking controller runs on the on-board PC. It is divided into low-level control, trajectory generation and stabilizing control. State estimation and a command server

9 http://www.sercos.com

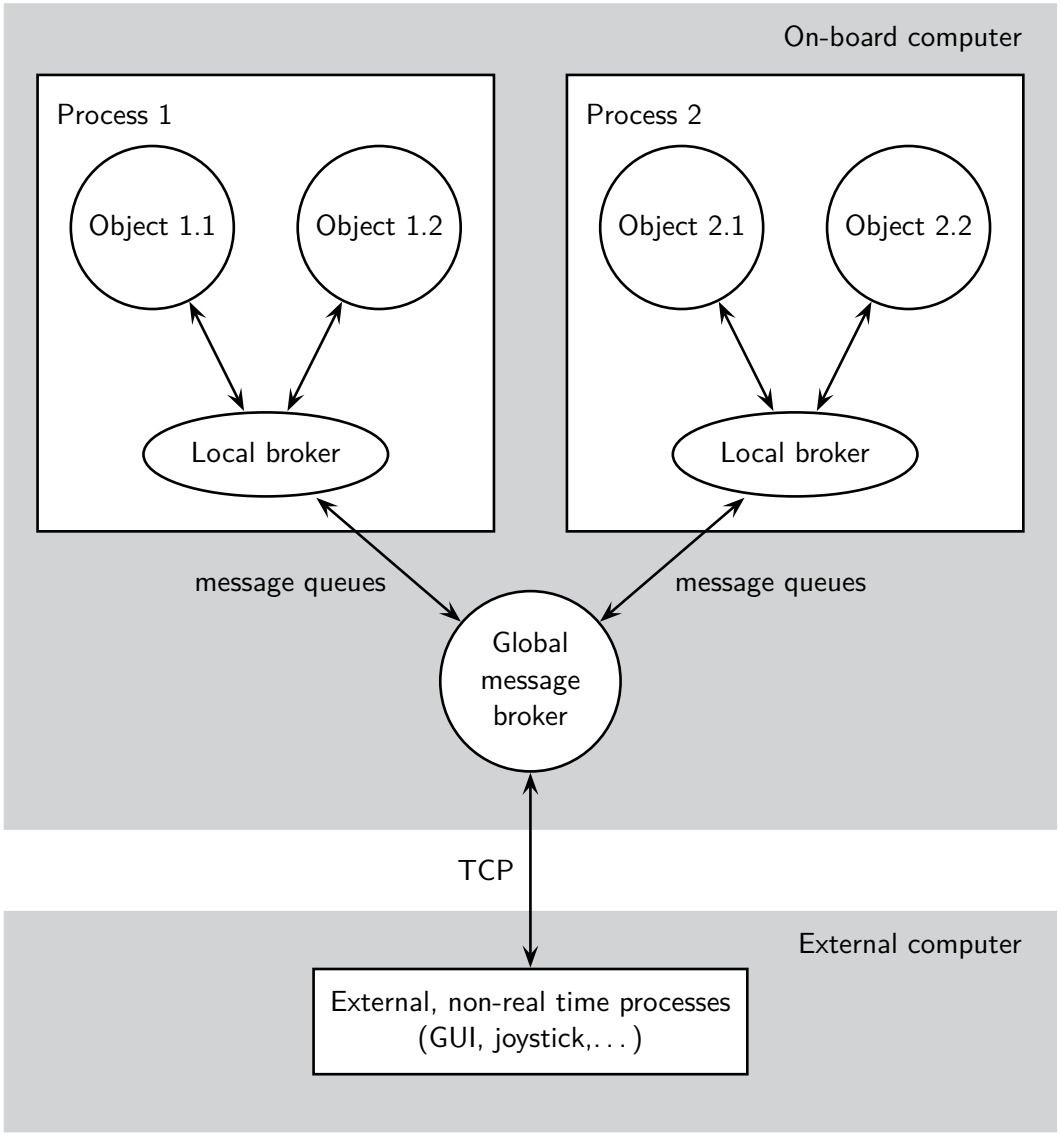


Figure 7.4: Schematic illustration of the publish/subscribe-based communication system

for the computer vision system also run on the on-board computer. All programs maintain a circular buffer of all relevant process data and controller variables. All data contains a 64-bit time-stamp which is generated by the low-level controller. This buffer is used for logging data during walking experiments and saved in the event of unexpected errors or program crashes.

A watchdog process disables the motor power supply if it does not receive a reset every few milliseconds. The low-level controller resets the watchdog every control cycle. In the event of an error, controllers can also notify the watchdog to disable the power supply.

Vision processing and user interfaces run on external computers, which are connected to the robot via Gigabit Ethernet.

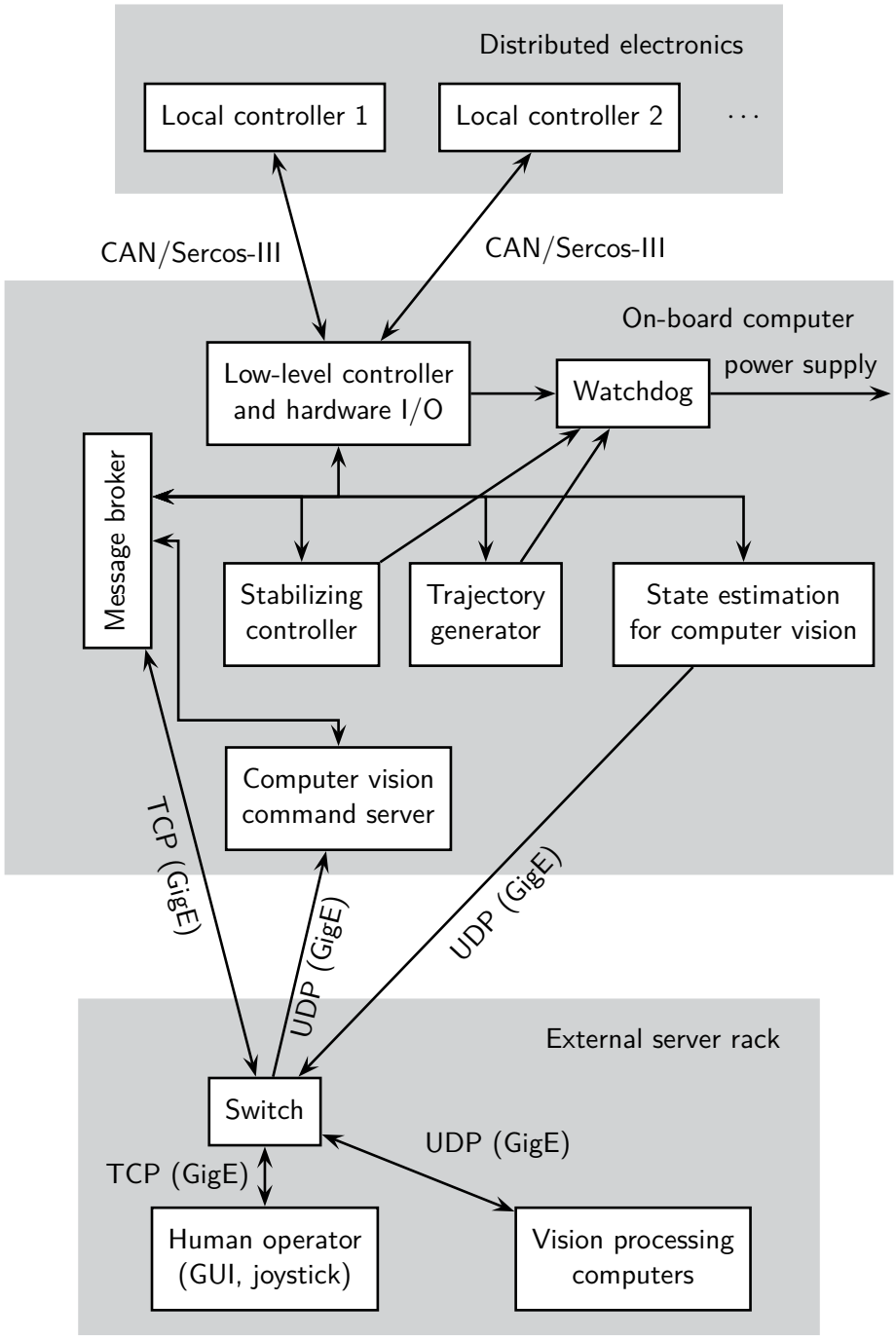


Figure 7.5: Overview of Lola's and Johnnie's real-time system

7.5 Chapter Summary

In this chapter, a brief survey of the software system developed for Johnnie and Lola was given. The design emphasizes modularity, code reuse, flexibility, efficiency and safety. The code is organized into libraries for physical models, mathematical tools, controllers and utilities. The run-time system consists of a number of controllers and supporting programs. This modularity increases both the flexibility and safety of the system. Besides the real-time controller and simulation programs, there are user interfaces for controlling the robot and applications for visualizing and analyzing data.

8 Results

8.1 Biped Walking

In this section, selected results from biped walking simulations and experiments are presented. Similar results were obtained for Johnnie and Lola, but the maximum walking speed for Lola is higher. All results shown in this section were obtained with Lola. Lola can walk forward, backwards, sideways and around corners. A subset of measurements for walking forward with 2.0 km/h and walking sideways with 0.7 km/h are shown and analyzed in the following. A more comprehensive set of measurements for walking forward with 0.0 km/h, 1.0 km/h, 2.0 km/h and 3.0 km/h and walking sideways with 0.7 km/h are given in Appendix G.

At the time of writing, Lola's maximum walking speed was 3.34 km/h. Figure 8.1 shows still images of Lola walking at this velocity. This speed was reached after less than three full weeks of walking experiments. The author believes that still higher walking speeds are possible with the same control system. In addition, a larger experimental area would simplify fast walking experiments. In the current setting, the robot must accelerate to its maximum speed in approximately four steps and decelerate just as rapidly.

According to publicly available data, Honda's Asimo walks at a maximum speed of 2.7 km/h[1] and only Petman [4] from Boston Dynamics walks faster at 7.1 km/h. However, this robot has non-humanoid leg kinematics and is not suitable for indoor use (cf. Section 1.2.2, p. 6). Toyota's running robot [116] and Honda's prototype robot [122] are significantly faster than Lola when running. Hubo-2 from KAIST can run at 3.3 km/h, while its walking speed is 1.4 km/h [80]. It is worth mentioning that all these robots are developed by companies with considerable financial support (Honda, Toyota and Boston Dynamics) or national research organizations (KAIST), not by universities. Summing up, Lola currently is the second fastest *walking* robot, the fastest *electrically actuated walking* robot and the fastest biped developed at a university.

8.1.1 Walking Forward

In this section, results from a walking experiment at 2 km/h are presented. The robot starts in a standing position, accelerates, walks for several steps at 2.0 km/h, decelerates and comes to a stop.

The upper body inclination error is shown in Figure 8.2. There is a small oscillation with a maximum amplitude of approximately 0.03 rad, synchronized with the step frequency.

1 http://asimo.honda.com/AsimoSpecs.aspx, accessed on 21/07/2010.

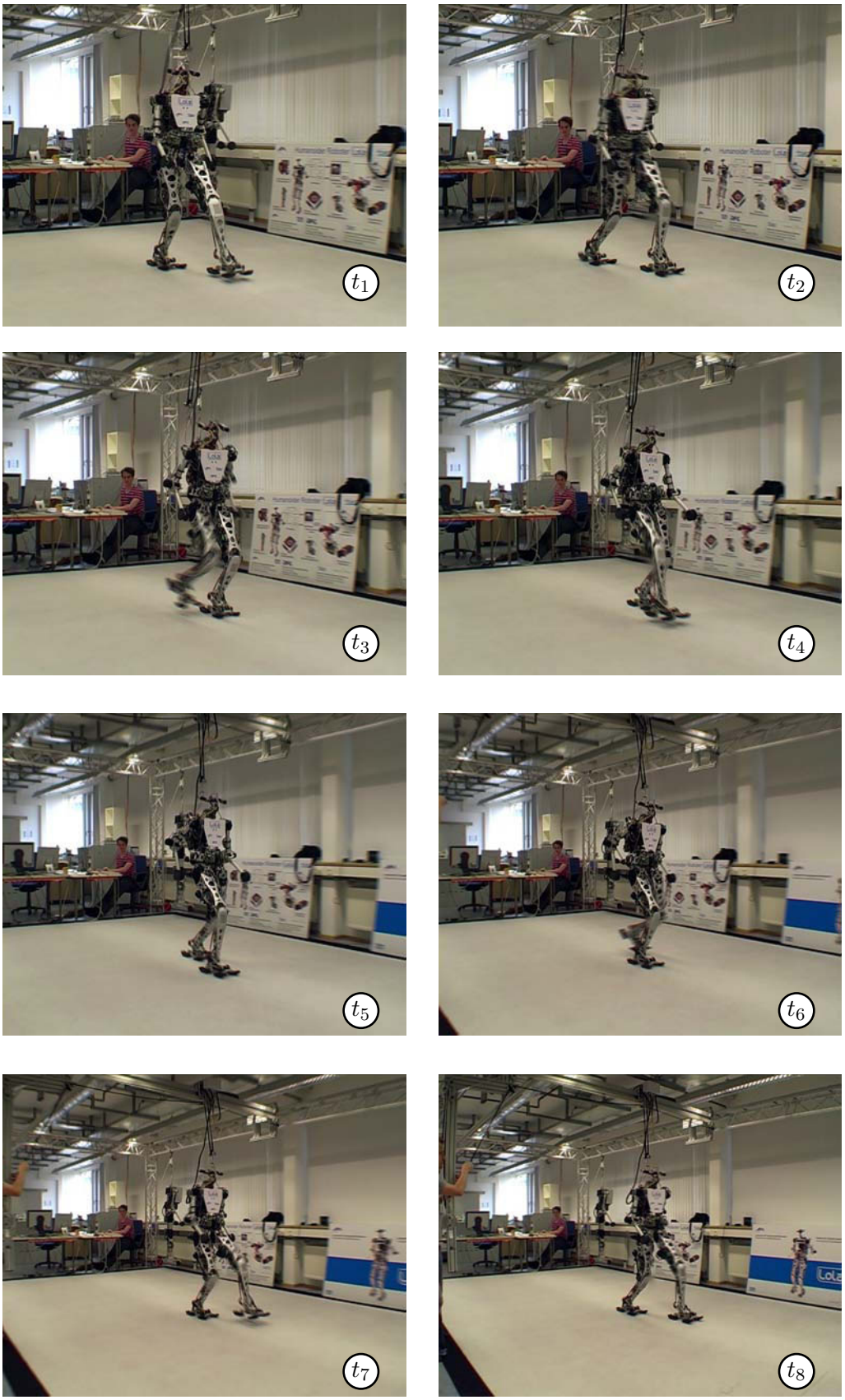


Figure 8.1: Lola walking at 3.34 km/h

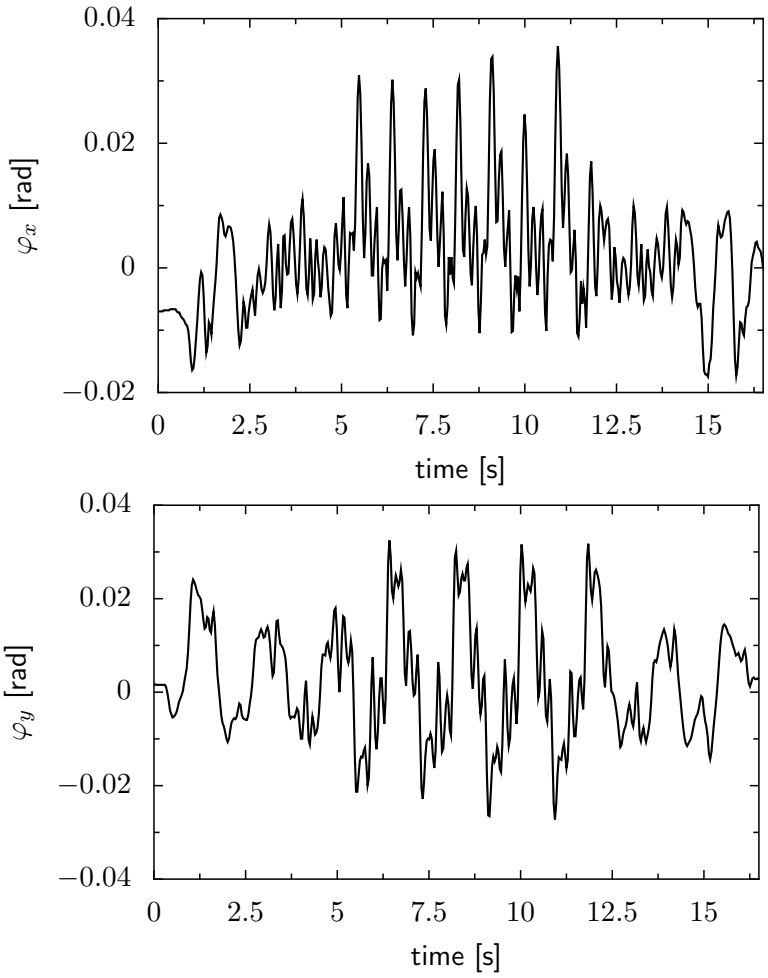


Figure 8.2: Upper body inclination error at 2 km/h

Figure 8.3 shows the measured z-component of the right force/torque sensor along with the reference value. The timing of ground contact and takeoff is very close to the reference values. The controller succeeds in preventing force peaks during initial contact. Note that the measured force is not exactly zero during the swing phase, due to inertial forces of the toe and heel segments and imperfect bias compensation.

Angles of the redundant toe and pelvis joints are shown in Figure 8.4. The toe joint is not used for the first and last steps, since these are only 30 cm long. During the middle of the experiment, when the robot is walking at 2 km/h, the toe and pelvis joints are used to increase the step length and reduce required joint velocities. For longer steps, the pelvis adduction angle is used to increase the height of the center of gravity without exceeding joint angle ranges.

Figure 8.5 shows the tracking error for the highly loaded knee and hip flexion joints. The tracking error is below $8 \cdot 10^{-3}$ rad. At the middle of the experiment, when the walking speed is highest, the tracking error increases due to higher loads, disturbance forces and angular velocities.

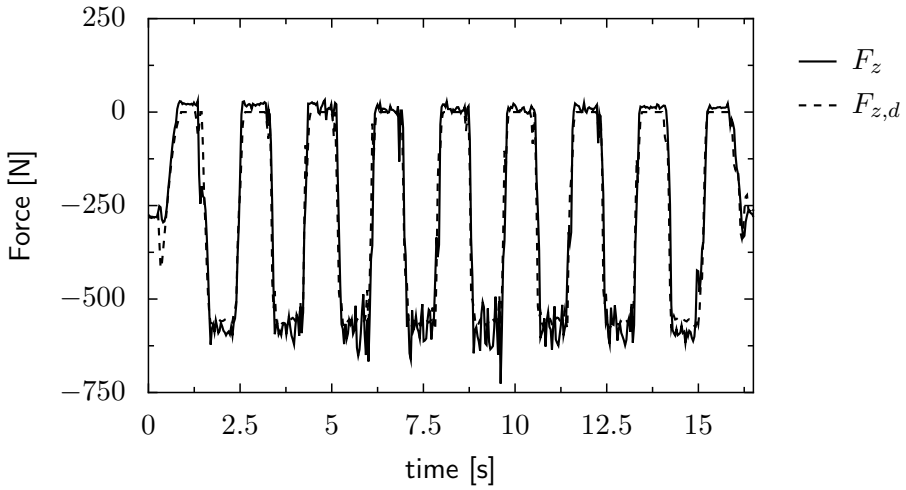


Figure 8.3: Reference ($F_{z,d}$) and actual (F_z) z-component of right force/torque sensor signal. The timing error between desired and actual support phases is very small.

8.1.2 Walking Sideways

Figure 8.6 shows still images of Lola walking sideways. The robot starts in a standing position, accelerates, walks sideways for several steps at 0.7 km/h, decelerates and comes to a stop. The upper body inclination error measured during the experiment is shown in Figure 8.7. There is a small oscillation with a maximum amplitude of approximately 0.025 rad, again, synchronized with the step frequency.

8.1.3 Comparison of Simulation and Measurement

In this section, results from dynamics simulations of Lola walking at 1 km/h are compared to measurements during walking experiments. The upper body inclination and contact forces are very sensitive to changes in the stiffness and damping of the foot-ground contact. This makes it difficult to get a perfect match between simulations and measurements. That being said, results obtained in simulations and experiments are very similar.

Force sensor signals and upper body inclination are shown in Figure 8.8 and Figure 8.9. There are only very small deviations in the vertical force component F_z. The difference in the torques T_x, T_y and the upper body inclination are larger. In both cases, however, the shape and the numerical values are quite similar. The difference for φ_x is mainly due to an offset of approximately $5 \cdot 10^{-3}$ rad, which might very well be a mounting misalignment of the IMU or a kinematic calibration error. For φ_y, the measured oscillations are larger than in the simulation. However, the absolute amplitudes are very small, most probably in the range of structural deformations not taken into account in the simulation. The measured torques have a similar profile, but are smaller than in the simulation.

Figure 8.10 shows joint angles for hip and ankle flexion. The hip flexion angle is virtually identical, while the measured ankle joint angle is a little larger than in the simulation. This is correlated with the smaller measured T_y. This indicates that the effective stiffness between ground contact and upper body is smaller in reality than

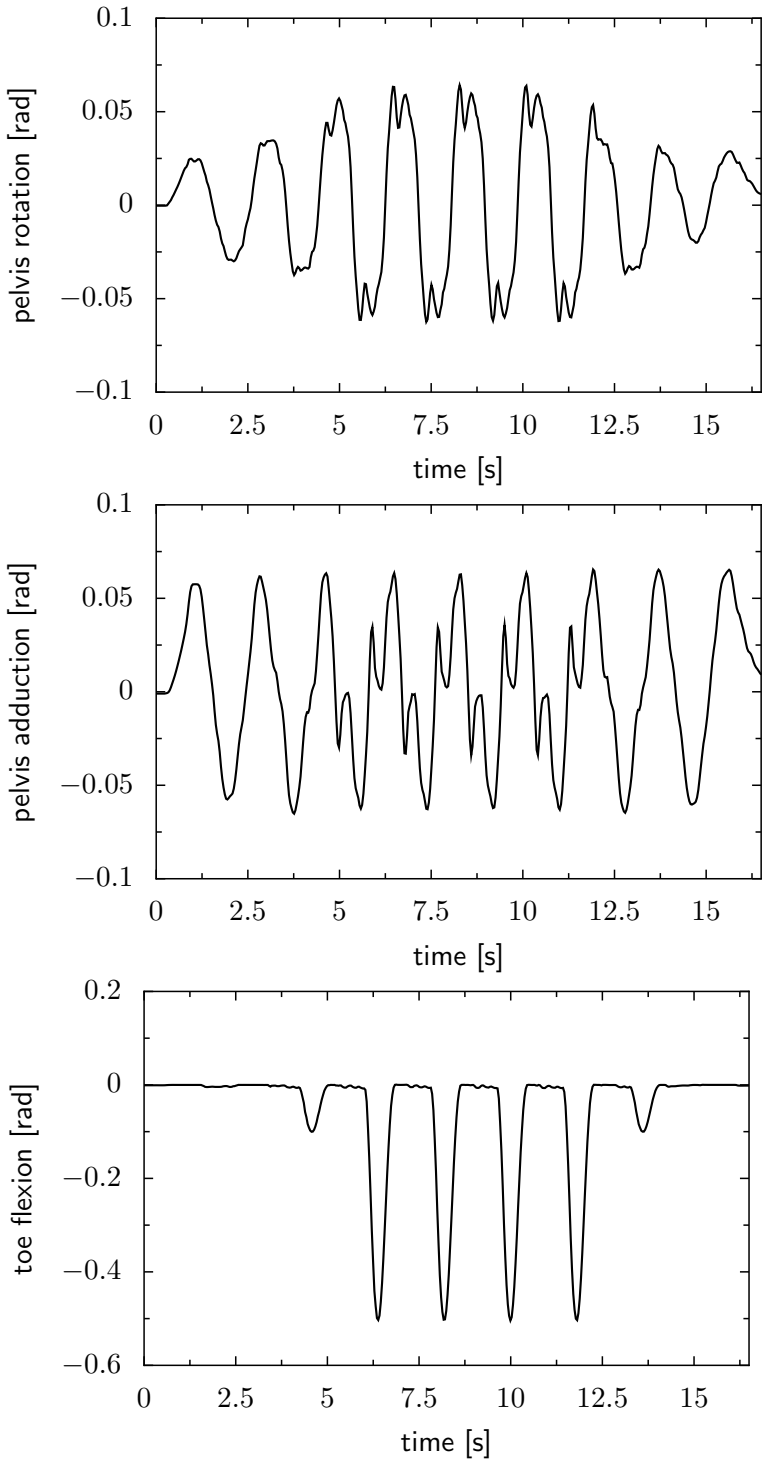


Figure 8.4: Pelvis and toe joint angles at 2 km/h

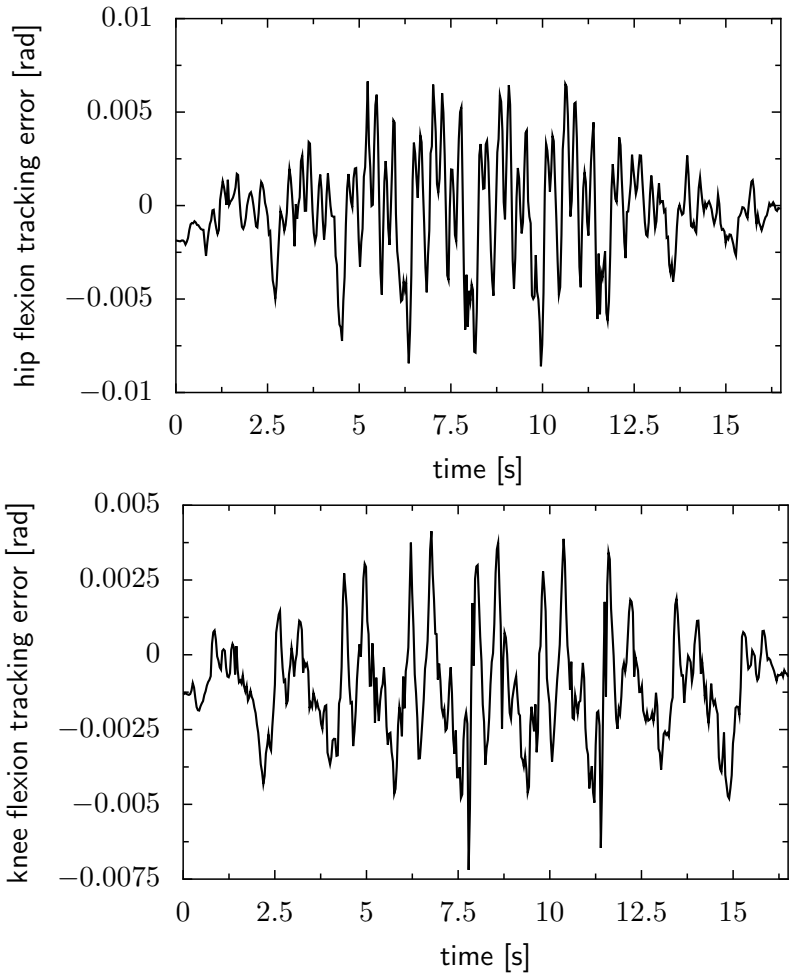


Figure 8.5: Joint angle tracking errors for the highly loaded hip flexion and knee joints at 2 km/h

in the robot models. This is probably due to the carpeting in the laboratory, which was not simulated, and the finite stiffness of bearings and links.

Figure 8.11 compares the simulated and measured tracking errors for the right hip adduction joint. There is a small difference at the beginning of the experiment, because the robot took several steps before the measurements began. Therefore the initial tracking error is approximately 10^{-3} rad, the same value as at the end of the experiment. In the simulation, the robot is put down onto the ground without taking any steps prior to the time interval shown. Consequently, this difference vanishes after the first step.

Overall, the simulation gives quite accurate results on all levels of system dynamics. Remaining differences can be attributed to different initial conditions in simulation and experiment and model simplifications. Including unmodeled effects such as bearing or ground compliance is possible. However, this would increase simulation times by orders of magnitude with only a small increase in accuracy. Moreover, determining correct parameters for, e.g., bearing compliance is difficult at best.

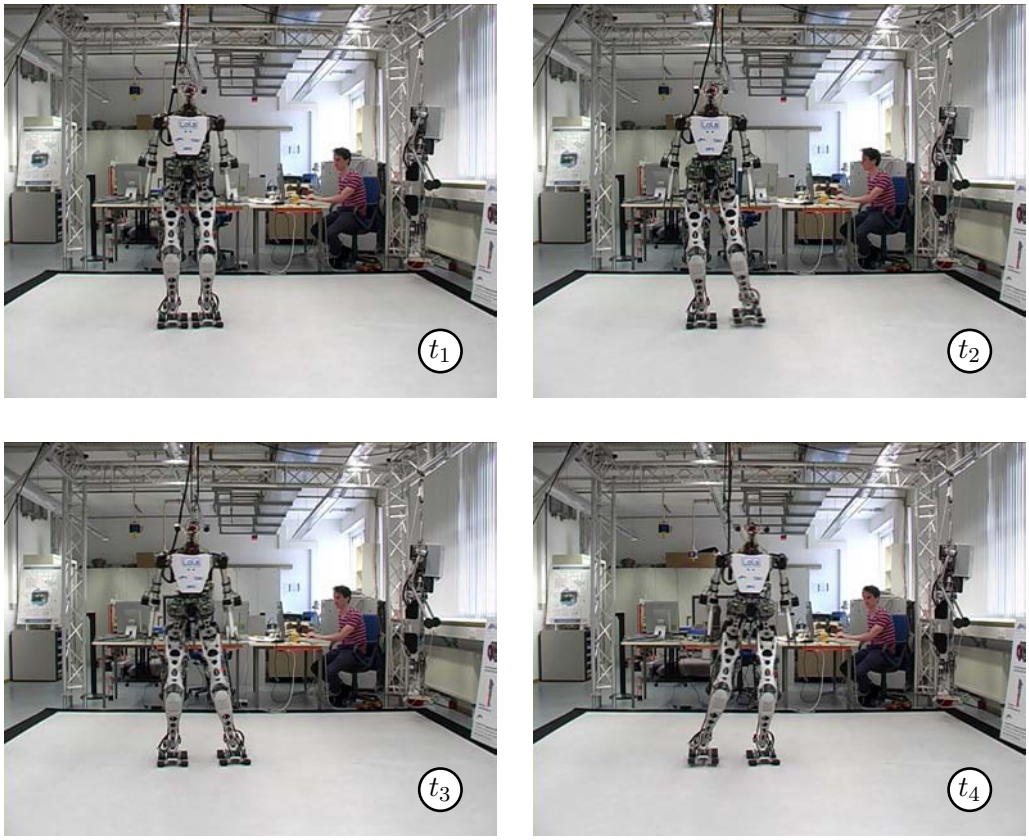


Figure 8.6: Lola walking sideways at 0.7 km/h

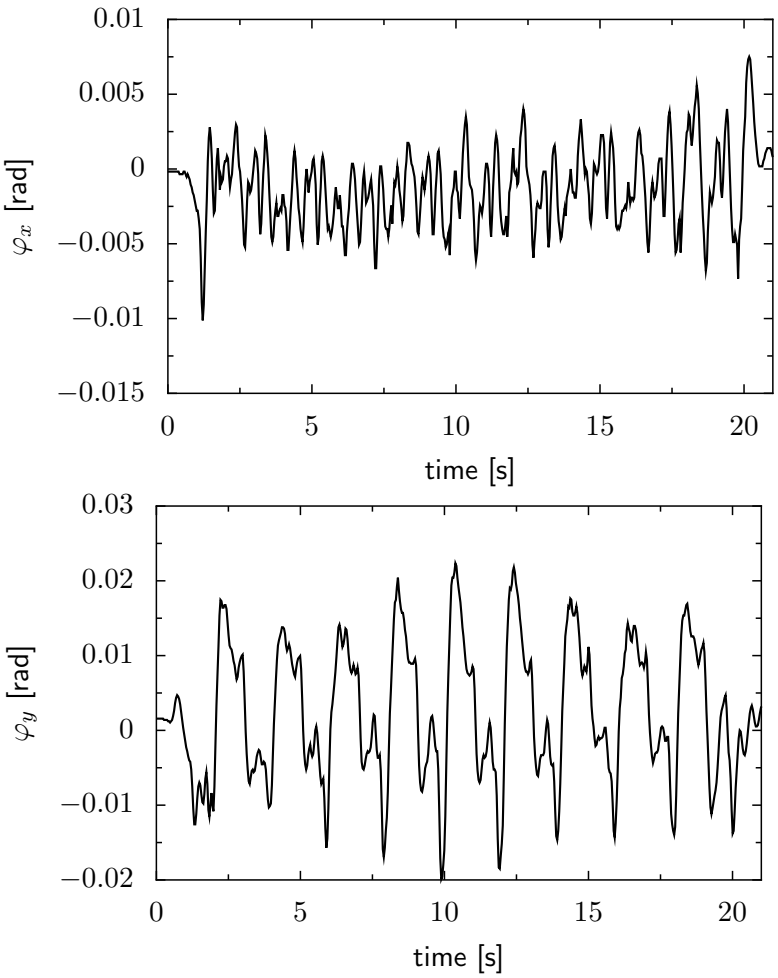


Figure 8.7: Upper body inclination error while walking sideways at 0.7 km/h

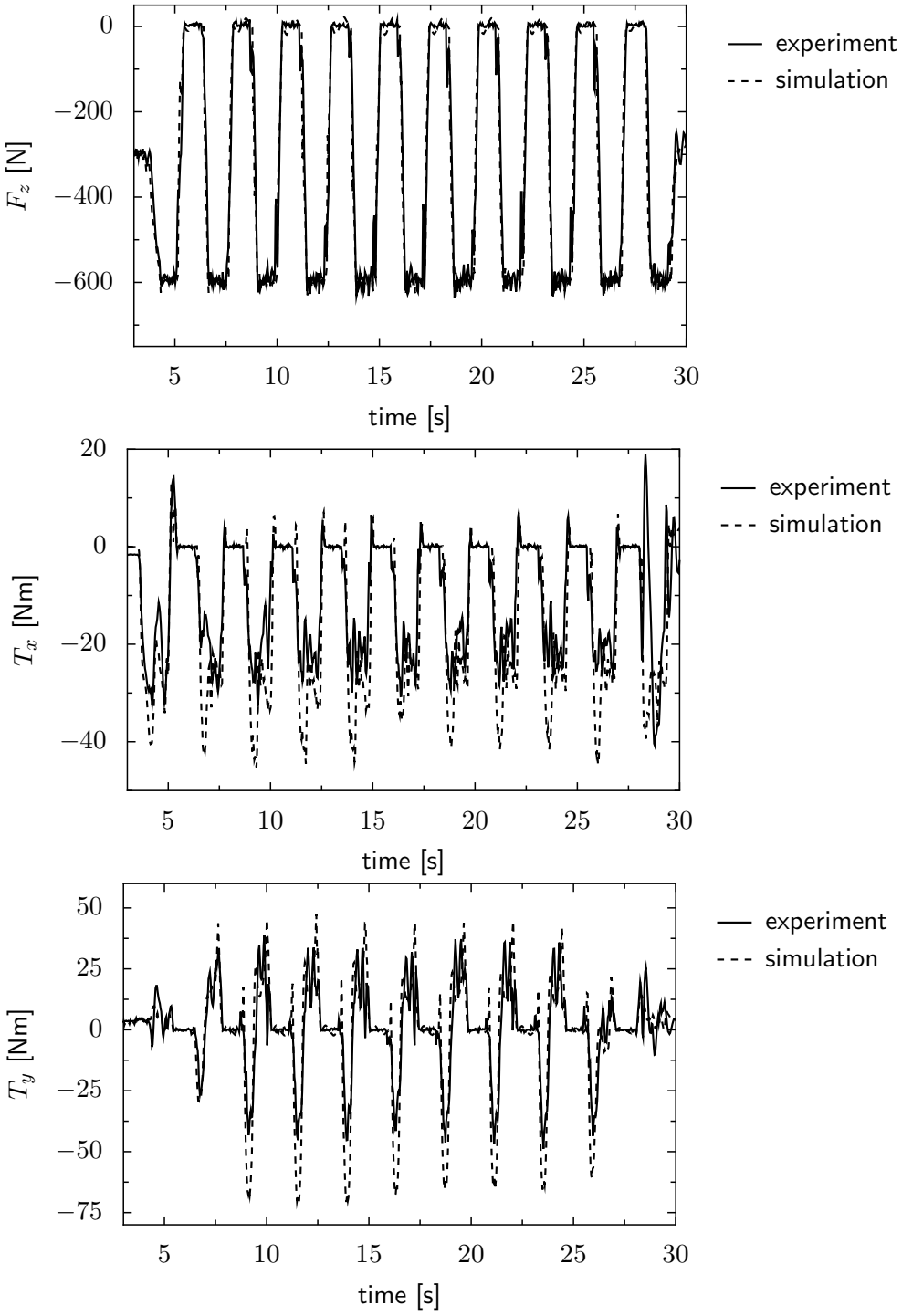


Figure 8.8: Comparison of measured and simulated force sensor signals

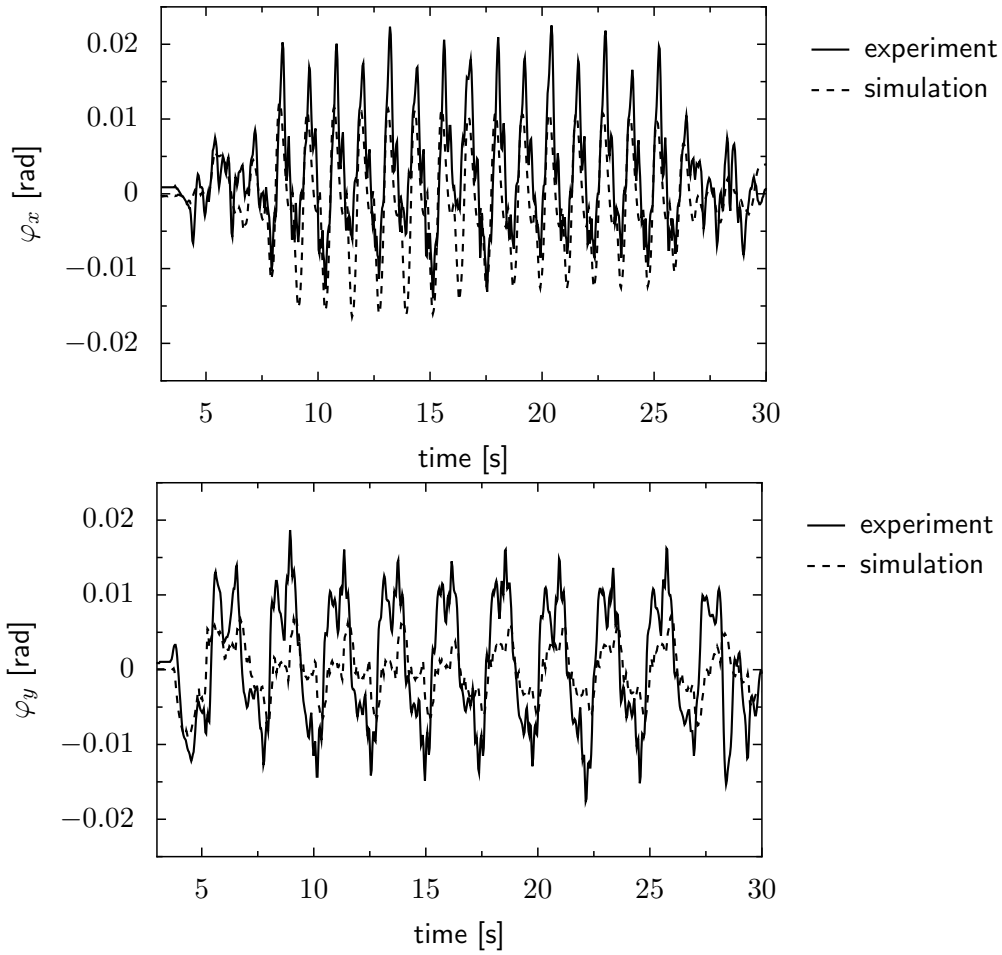


Figure 8.9: Comparison of measured and simulated IMU signal

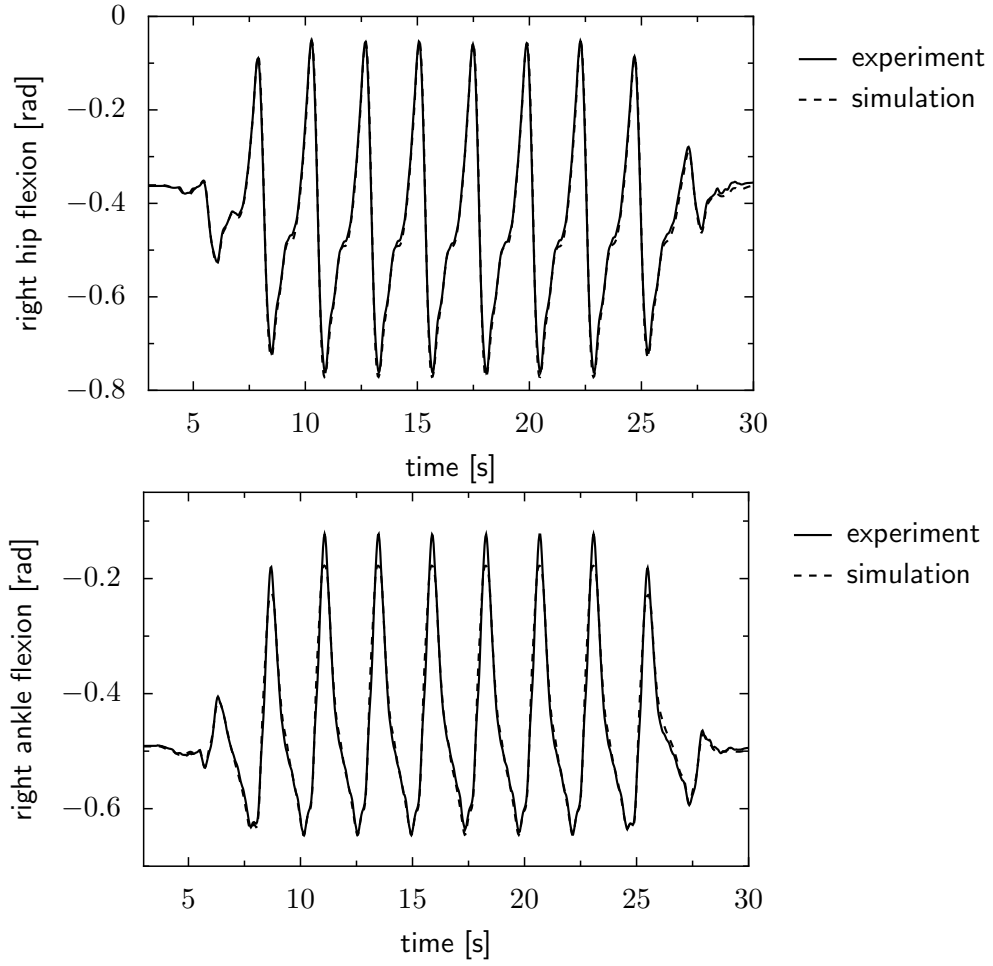


Figure 8.10: Comparison of measured and simulated joint angles

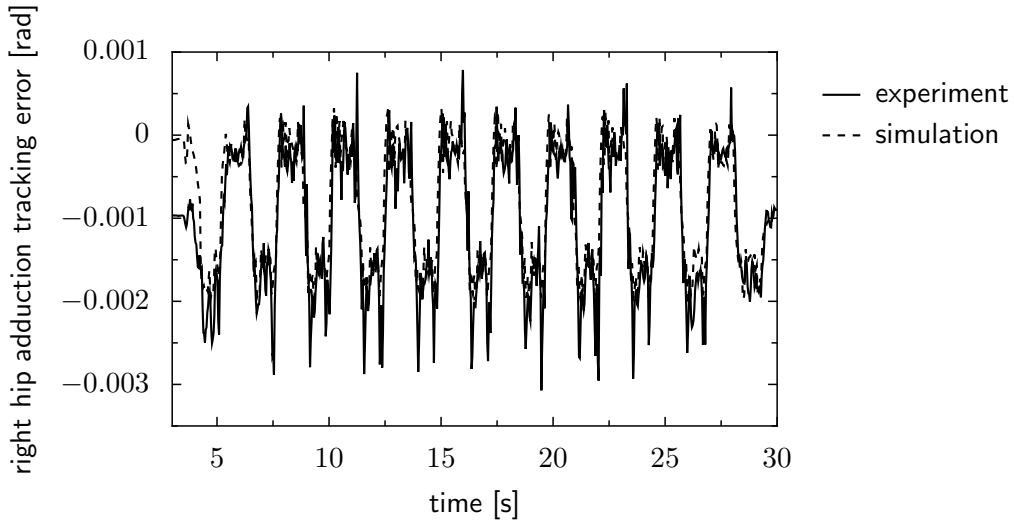


Figure 8.11: Comparison of measured and simulated joint angle tracking error

8.2 Autonomous Locomotion

Lola's capabilities of autonomous locomotion were publicly demonstrated at the *Hannover Messe 2010*[2] in more than 25 presentations. The robot autonomously explored a $4 \times 7\,\mathrm{m}^2$ area while avoiding unknown objects of various shapes and sizes. Such an experiment demonstrates the capabilities of both computer vision *and* walking control system.

Contrary to periodic walking or executing a fixed sequence of walking steps, autonomous locomotion requires quick reactions to a changing environment in order to avoid collisions with obstacles. This was enabled by the real-time trajectory generation system, which is capable of quickly generating feasible walking trajectories (cf. Chapter 4). During reactive navigation among obstacles, the robot continuously accelerates and decelerates, changes its walking direction, speed and the curvature of its path. This is very demanding for both trajectory generation and stabilizing control, since there is an implicit assumption in the trajectory generator that the robot will eventually start a periodic gait (cf. Section 4.6). Moreover, conditions at a trade fair are less perfect than in a laboratory. At the *Hannover Messe 2010*, the tile flooring below the carpet was slightly uneven and there were a number of small holes with an area of up to $10 \times 3\,\mathrm{cm}^2$ and a depth of 1-2cm.

The robot successfully navigated among large obstacles such as tables and chairs (cf. Figure 8.12(a)), which were placed at arbitrary locations and moved around during presentations. To demonstrate the fact that the robot does not use prior knowledge about the obstacles, objects such as bags or jackets provided by the visitors were placed in front of the robot. Thanks to the use of high-resolution cameras, the robot is capable of detecting even small objects such as cell phones or scissors (cf. Figure 8.12(c)). Lola's capability of navigating in dynamic environments was demonstrated by participating researchers walking in front of the robot (cf. Figure 8.12(b)).

2 The Hannover Messe is the world's largest industrial trade fair (`http://www.hannovermesse.de`).

(a) Lola navigating among furniture

(b) Lola avoiding a human being

(c) Lola avoiding a small object (scissors)

Figure 8.12: Examples of autonomous navigation among unknown obstacles

9 Conclusion

Human-like robots have great potential as general-purpose service robots. Because of their humanoid morphology, their use requires no changes to existing infrastructure and the human-like appearance simplifies human-machine communication and enhances the chances of acceptance from a psychological point of view. One of the most basic requirements for such robots is reliable, fast and autonomous biped locomotion. The preceding chapters of this thesis presented a framework for simulating and controlling biped walking robots. A brief summary of key ideas and contributions is given in the following section. Section 9.2 contains a concluding discussion and the final section outlines suggestions for future research on biped locomotion.

9.1 Summary

This thesis covers the simulation and control of biped walking robots. In the first part, a modular system for simulating such robots was presented. It is designed as a library of component models, contact and ordinary differential equation solvers that are combined to problem-specific multibody simulation programs. The library contains models for the dynamics of tree-structured multibody systems, nonlinear drive kinematics and electrical motor dynamics. The dynamics of harmonic drive gears are taken into account by an improved model for load and speed dependent friction and a nonlinear model of gear elasticity. Compliant, unilateral contacts can be modeled using numerically efficient, decoupled spring-damper systems or linear finite element models. Arbitrary polygonal environments are considered, enabling simulations of uneven ground, obstacle avoidance and stair climbing.

Because of the modular design, problem-specific simulations with different balances between modeling depth and simulation time can be constructed. At the one end of the spectrum, a reduced order model enables (near) real-time simulation of global dynamics in order to quickly assess the performance of trajectory generation and stabilizing control. At the other end, a simulation including all component models enables a detailed analysis of system dynamics. Such detailed simulations were used for calculating loads, required motor torques and other fundamental design data for Lola. In an iterative process, the hardware design was improved by Lohmeier [62] to obtain a modified robot model, which was then used to assess the performance improvements and calculate design data for the next iteration. Simulations show good correspondence to data measured during walking experiments.

The second part of the thesis presents a hierarchical real-time walking control system suitable for fast and autonomous biped walking. The gait is coordinated by a finite state machine that selects and/or configures planning and control modules according to the current robot state. At the highest level, a sequence of steps is planned that satisfies the commands input by a human operator or a computer vision

system. At this level, a reactive step sequence planner can avoid small obstacles by choosing parameters for stepping over them. The step sequence is used to generate constraints and boundary conditions for the calculation of reference trajectories in workspace coordinates. A new method for calculating center of gravity trajectories based on spline collocation was developed. The method enables fast calculation of reference trajectories for an enhanced inverted pendulum model with swing-leg disturbance terms. Contrary to the most widespread prior real-time methods, a non-constant height of the center of gravity can be taken into account.

Simply tracking joint angles for ideal reference trajectories quickly leads to instability, because the compliant unilateral contact makes the system very sensitive to modeling errors and disturbances. A new method for stabilizing biped walking by modifying the reference trajectories was developed. The basic approach is to control contact forces in order to stabilize unactuated degrees of freedom. Desired contact forces and task-space trajectories are tracked by a hybrid position/force control. A new formulation of hybrid position/force control for biped robots based on an inner position control loop is presented. By using an explicit contact model, arbitrary and varying foot shapes can be taken into account. In addition, deviations from the reference trajectories required for force control can be mapped into arbitrary task-space dimensions, as long as these are not orthogonal to the forces to be controlled.

Lola is redundant with respect to the chosen task space coordinates. The framework proposed by Liégeois is adopted for local optimization of redundancy and joint limit avoidance. Decoupled, high-gain joint position controllers are used at the lowest level of the hierarchy.

Using this system, Lola reached a maximum walking speed of 3.34 km/h in experiments. When combined with a computer vision system developed by the Autonomous Systems Technology Institute, Universität der Bundeswehr, Lola is also capable of autonomously navigating among unknown obstacles.

9.2 Discussion

Lola's maximum walking speed currently makes it the fastest biped developed at a university and the fastest electrically driven walking humanoid (cf. Section 8.1). However, comparing the walking speed of humanoid robots is difficult, because there are significant differences in the overall height, mass distribution, actuation and electronics and the achievable walking speed does not scale linearly with the size of the robot. Nevertheless, it seems reasonable to state that Lola's walking performance is at the cutting edge of current walking technology. Compared to a human being, however, the walking speed is still quite moderate.

Walking is fairly robust to disturbances such as small obstacles of approximately two cm height or moderate pushes. Still, it is quite possible to destabilize the system with, e.g., large pushes during slow walking or smaller pushes during very fast walking.

Thanks to the new trajectory planning and control algorithms, faster acceleration and deceleration of the robot could be achieved. Still, very large accelerations and changes in the walking direction can destabilize the machine. The capability of quickly reacting to unexpected obstacles or disturbances by using large compensating

motions makes biped walking of humans very versatile and reliable. This is an area in which current biped walking technology is still lacking.

A further limitation concerns motions close to the kinematic limits such as stepping over very large obstacles or taking extremely long steps. Since foot and center of gravity trajectories are planned in task-space, it is difficult to take such kinematic constraints into account. Unfortunately, planning trajectories for such motions requires taking the full kinematics of the robot into account at the planning stage, which is currently not possible in real-time.

9.3 Recommendations for Future Research

From the experience gained during this thesis, several suggestions for future research on biped walking robots can be made.

Step Sequence Planning

Local step sequence planning could be improved by not only adapting the step length, but also the walking direction and not only stepping over, but also onto obstacles. This could be achieved by using, e.g., $A^\star$ search for finding an optimal step sequence for the next few steps.

Trajectory Generation

The trajectory generation method presented here enables the use of arbitrary trajectories for the vertical center of gravity position z_{CoG}. Preliminary results suggest that significant gains in performance and stability can be achieved by using a non-constant z_{CoG}-trajectory. Therefore, development of a systematic method for generating such trajectories is suggested.

Stabilizing Control

While the proposed walking controller is quite robust, large disturbances destabilize the robot because the control authority is effectively limited by the unilateral ground contact. The gait cycle can be stabilized by modifying the landing position of the foot. This approach has been successfully used in various bipeds, from Raibert's hopping robots to Honda's Asimo. Compared to these robots, Johnnie's and Lola's ratio of upper body mass to leg mass is significantly lower. Therefore, predicting the effects of modified foot landing positions is more difficult, complicating the application of this control method. Nevertheless, the potential stability and performance gains justify following this approach and faster computers should make it feasible in the near future.

Low-Level Control

While the decoupled joint position control performs very well at low and moderate walking speeds, tracking errors increase for very fast motions because of increased dynamic coupling and gear friction.

Currently, the maximum sampling frequency of the controller is limited to approximately 3.5 ms by the CAN-bus system. Simulations suggest that the performance of both joint position and force control could be improved considerably by increasing

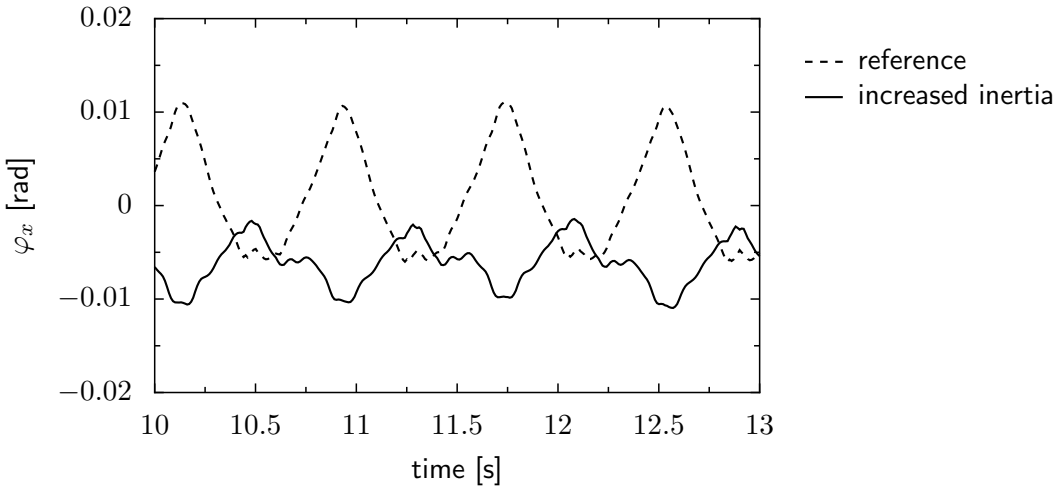


Figure 9.1: Increased upper body inertia could be used to reduce upper body oscillations. The black line shows a simulation with a thin steel torus of 0.5 m diameter weighing 10 kg added to the upper body. No other changes were made to the model or the controller.

the sampling frequency to 1 kHz. This will be possible when the Sercos-III system currently being implemented becomes operational.

Better tracking could also be achieved by including friction models and multi-body dynamics into the low-level joint position control. As has been suggested by Lohmeier [62], force sensing could be integrated into Lola's ankle and knee drives. This could be used to significantly improve link-side position control by using full-state feedback for the elastic joint. In addition, such sensors would enable direct force control with a significantly higher bandwidth than the current contact force control.

Learning Robot and Environment Model Parameters

The model-based approach makes the system sensitive to systematic errors in robot and environment models. Most errors, however, are due to parameter uncertainty rather than structural errors in system models. Learning model parameters on-line using on-board sensing would increase model fidelity and enable an automatic adaptation, e.g., to changing ground stiffness and inclination.

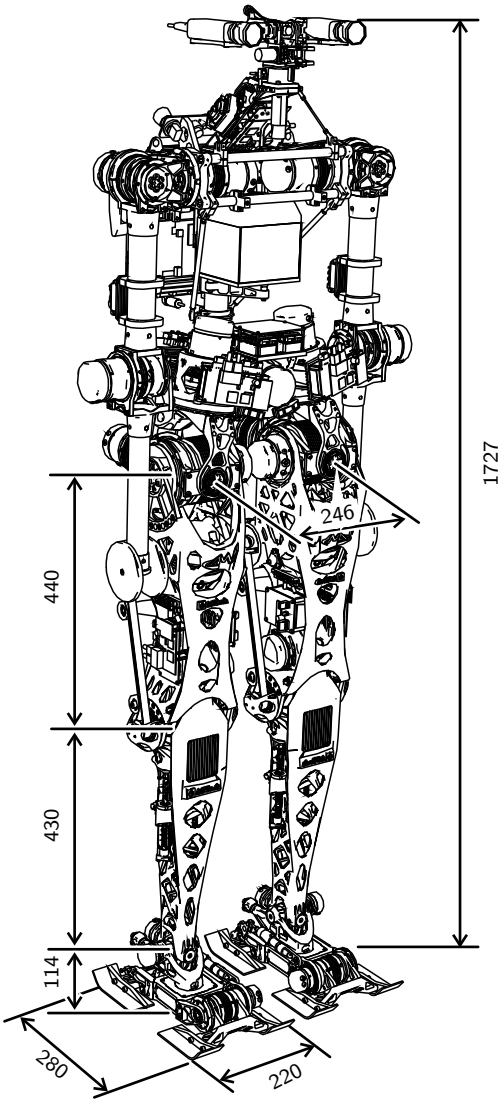
Hardware

As noted above, Lola's upper body mass and inertia are very low compared to a human being or other biped robots. Simulations show that upper body oscillations during periodic walking can be reduced significantly by increasing upper body inertia, as shown in Figure 9.1. At the same time, higher inertia also increases the forces required for stabilizing the robot. Moreover, higher mass and inertia reduce the acceleration capabilities. A detailed study of this issue is proposed, in order to obtain an optimal choice for the upper body inertia.

Finally, the importance of contact dynamics for walking stability suggests further research on the design of robotic feet. This research should be tightly integrated with efforts to optimally exploit foot dynamics in the walking controller.

Appendix A

Lola's Basic Technical Data



Mass	60 kg
Height	1.73 m
Active DoFs	25
Force sensing	2 6-axis force/torque sensors
Inertial sensing	Orientation and angular velocity estimation with 3 FOG and 3 MEMS accelerometers
Position sensing	25 incremental encoders (motor side) and 22 absolute encoders (link-side, all except head joints)
Actuators	Permanent magnet synchronous motors

Detailed information on Lola's hardware is given in [62], a detailed description of Johnnie's hardware can be found in [25, 60].

Appendix B

Multibody System Topology of Lola

The basic topology of Lola's multibody model is listed in the following table (motors not shown for clarity).

Body	Parent	Mass [kg]
Torso	None	$9.886 \cdot 10^{0}$
Left shoulder flexion	Torso	$5.610 \cdot 10^{-1}$
Left shoulder adduction	Left shoulder flexion	$1.900 \cdot 10^{0}$
Left elbow flexion	Left shoulder adduction	$1.486 \cdot 10^{0}$
Right shoulder flexion	Torso	$5.610 \cdot 10^{-1}$
Right shoulder adduction	Right shoulder flexion	$1.891 \cdot 10^{0}$
Right elbow flexion	Right shoulder adduction	$1.486 \cdot 10^{0}$
Pelvis rotation	Torso	$1.316 \cdot 10^{0}$
Pelvis adduction	Pelvis rotation	$6.283 \cdot 10^{0}$
Left hip rotation	Pelvis adduction	$1.844 \cdot 10^{0}$
Left hip adduction	Left hip rotation	$2.838 \cdot 10^{0}$
Left hip flexion	Left hip adduction	$6.419 \cdot 10^{0}$
Left knee flexion	Left hip flexion	$3.403 \cdot 10^{0}$
Left ankle adduction	Left knee flexion	$1.380 \cdot 10^{-1}$
Left ankle flexion	Left ankle adduction	$1.643 \cdot 10^{0}$
Left toe flexion	Left ankle flexion	$7.970 \cdot 10^{-1}$
Right hip rotation	Pelvis adduction	$1.844 \cdot 10^{0}$
Right hip adduction	Right hip rotation	$2.838 \cdot 10^{0}$
Right hip flexion	Right hip adduction	$6.503 \cdot 10^{0}$
Right knee flexion	Right hip flexion	$3.403 \cdot 10^{0}$
Right ankle adduction	Right knee flexion	$1.380 \cdot 10^{1}$
Right ankle flexion	Right ankle adduction	$1.648 \cdot 10^{0}$
Right toe flexion	Right ankle flexion	$7.970 \cdot 10^{-1}$
Head pan	Torso	$2.490 \cdot 10^{-1}$
Head tilt	Head pan	$1.013 \cdot 10^{0}$

Appendix C

Harmonic Drive Friction Model Parameters

Friction Model Parameters

Parameters for friction model (2.61) for the major gears used in Lola are given in the following table.

Table C.1: Harmonic Drive gear friction model parameters

Type	b	T_0	μ	γ	N
HFUC-11-100	$6.63 \cdot 10^{-6}$	$7.75 \cdot 10^{-3}$	0.0	$6.59 \cdot 10^{-4}$	100
HFUC-14-100	$1.04 \cdot 10^{-5}$	$1.21 \cdot 10^{-2}$	0.0	$6.59 \cdot 10^{-4}$	100
HFUC-17-100	$3.20 \cdot 10^{-5}$	$3.72 \cdot 10^{-2}$	0.0	$6.59 \cdot 10^{-4}$	100
HFUC-20-100	$5.33 \cdot 10^{-5}$	$6.20 \cdot 10^{-2}$	0.0	$6.58 \cdot 10^{-4}$	100
HFUC-25-100	$8.94 \cdot 10^{-5}$	$1.04 \cdot 10^{-1}$	0.0	$6.58 \cdot 10^{-4}$	100
HFUC-32-50	$1.52 \cdot 10^{-4}$	$2.46 \cdot 10^{-1}$	$5.49 \cdot 10^{-3}$	$6.21 \cdot 10^{-4}$	50

Friction Model Comparison

In this section the efficiencies of harmonic drives taken from catalog data are compared to predictions by the implemented friction model (2.61).

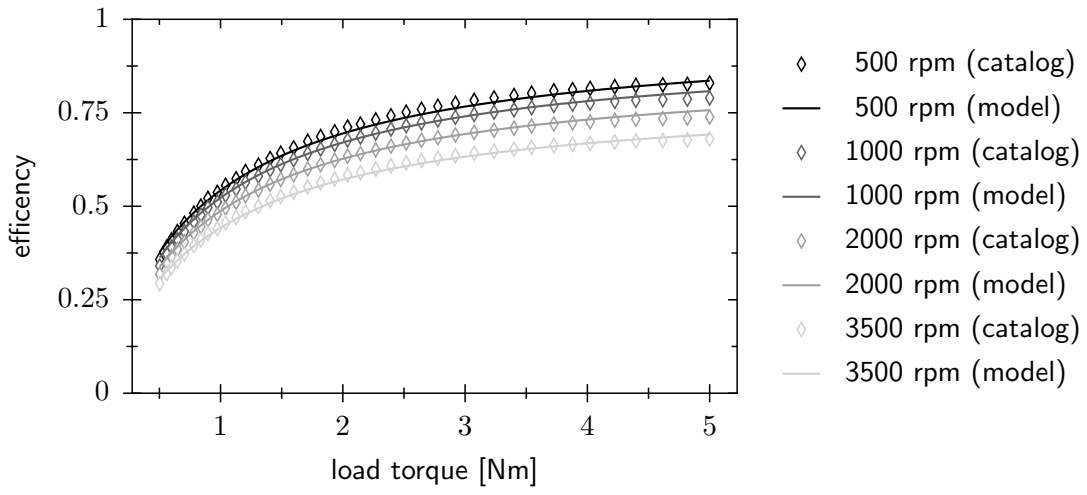


Figure C.1: Harmonic drive gear HFUC-11-100

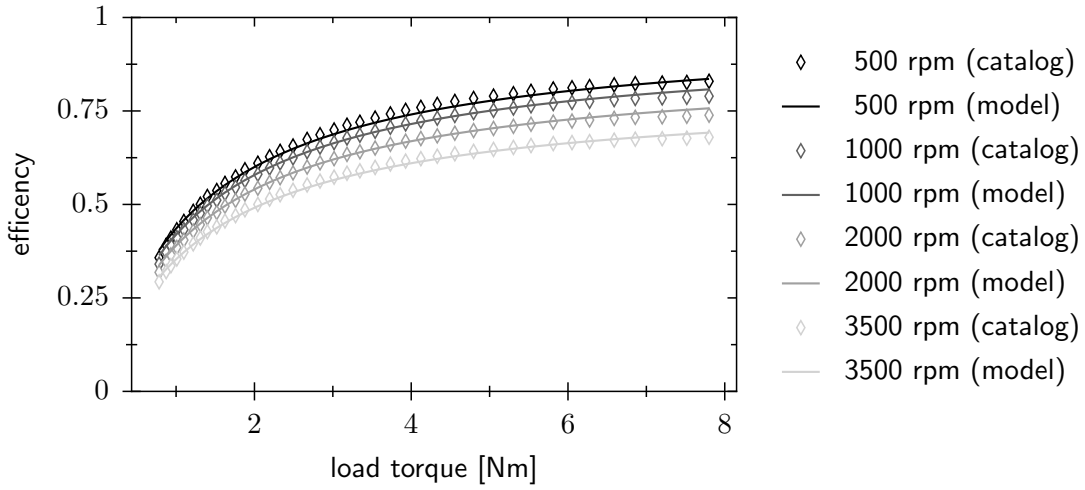


Figure C.2: Harmonic drive gear HFUC-14-100

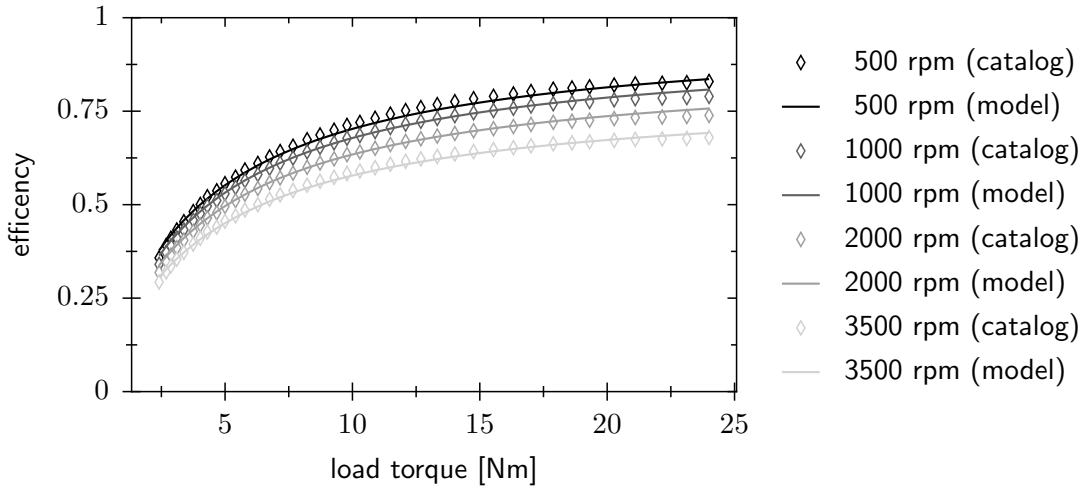


Figure C.3: Harmonic drive gear HFUC-17-100

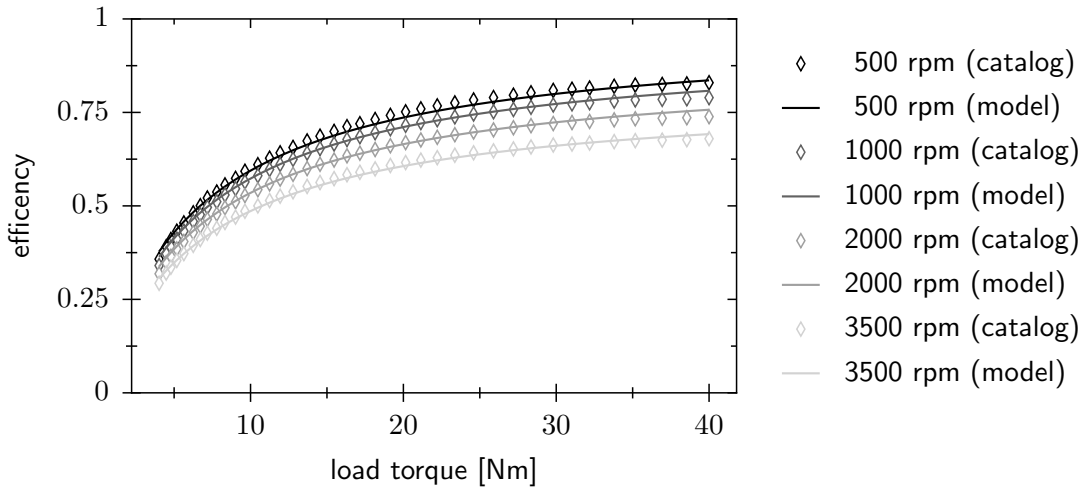


Figure C.4: Harmonic drive gear HFUC-20-100

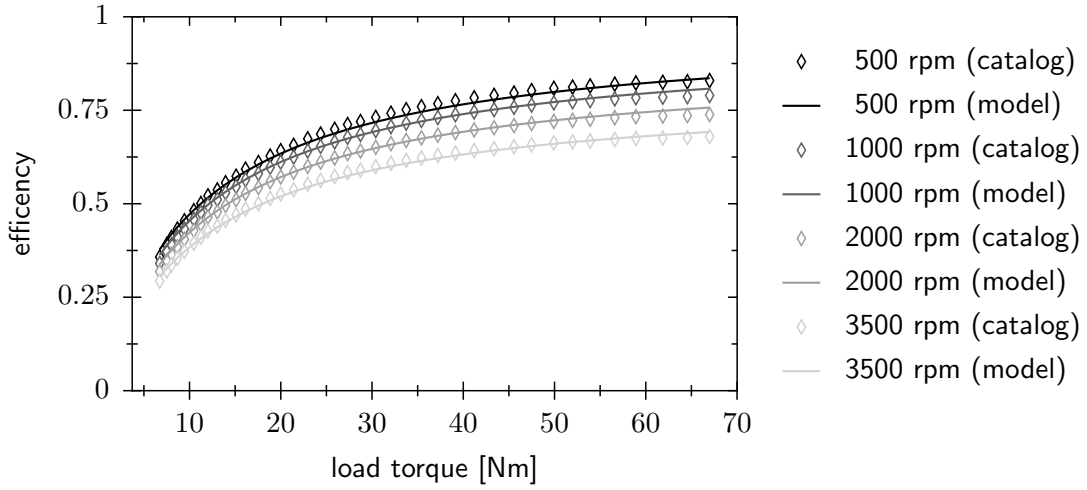


Figure C.5: Harmonic drive gear HFUC-25-100

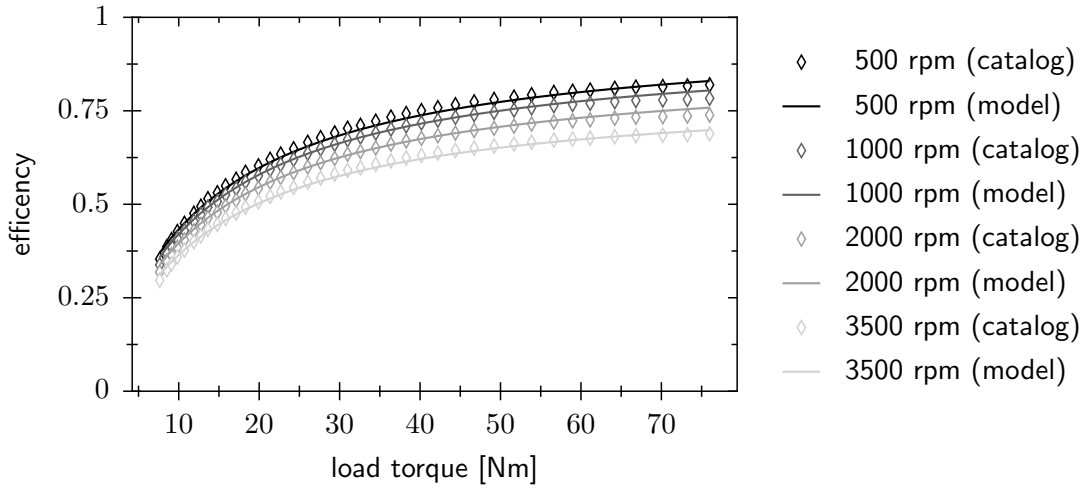


Figure C.6: Harmonic drive gear HFUC-32-50

Appendix D

Upper Body Kinematics

This appendix lists the kinematic equations for calculating upper body rotation matrices from Euler angles or the spatial representation used for real-time control.

The transform matrix $\boldsymbol{A}_{uI}$ from the inertial coordinate system into the upper body coordinate system is given by:

$$\boldsymbol{A}_{uI} = \begin{pmatrix} \mathrm{c}\,\psi\,\mathrm{c}\,\varphi - \mathrm{s}\,\psi\,\mathrm{c}\,\vartheta\,\mathrm{s}\,\varphi & \mathrm{s}\,\psi\,\mathrm{c}\,\varphi + \mathrm{c}\,\psi\,\mathrm{c}\,\vartheta\,\mathrm{s}\,\varphi & \mathrm{s}\,\vartheta\,\mathrm{s}\,\varphi \\ -\mathrm{c}\,\psi\,\mathrm{s}\,\varphi - \mathrm{s}\,\psi\,\mathrm{c}\,\vartheta\,\mathrm{c}\,\varphi & -\mathrm{s}\,\psi\,\mathrm{s}\,\varphi + \mathrm{c}\,\psi\,\mathrm{c}\,\vartheta\,\mathrm{c}\,\varphi & \mathrm{s}\,\vartheta\,\mathrm{c}\,\varphi \\ \mathrm{s}\,\psi\,\mathrm{s}\,\vartheta & -\mathrm{c}\,\psi\,\mathrm{s}\,\vartheta & \mathrm{c}\,\vartheta \end{pmatrix} \tag{D.1}$$

Here c and s are used to denote respectively sin and cos.

Assuming that the upper body is inclined by less than $\pi/2$, the Euler angles for a given transform matrix can be calculated using the following equations:

$$\psi = \mathrm{atan2}(\boldsymbol{A}_{TI,2,0}, -\boldsymbol{A}_{TI,2,1}) \tag{D.2}$$

$$\vartheta = \mathrm{acos}(\boldsymbol{A}_{TI,2,2}) \tag{D.3}$$

$$\varphi = \mathrm{asin}(\boldsymbol{A}_{TI,0,2} / \sin\vartheta) \tag{D.4}$$

The rotation matrix from inertial reference frame into the upper body planning frame is given by:

$$\boldsymbol{A}_{TI} = \begin{pmatrix} \mathrm{s}\,\varphi_x\,\mathrm{c}\,\alpha & -\mathrm{s}\,\alpha\,\mathrm{s}\,\varphi_y & \mathrm{s}\,\varphi_x\,\mathrm{s}\,\alpha\,\mathrm{c}\,\varphi_y - \mathrm{c}\,\varphi_x\,\mathrm{s}\,\varphi_y\,\mathrm{c}\,\alpha \\ \mathrm{s}\,\varphi_x\,\mathrm{s}\,\alpha & \mathrm{c}\,\alpha\,\mathrm{s}\,\varphi_y & -\mathrm{c}\,\varphi_x\,\mathrm{s}\,\varphi_y\,\mathrm{s}\,\alpha - \mathrm{s}\,\varphi_x\,\mathrm{c}\,\varphi_y\,\mathrm{c}\,\alpha \\ \mathrm{c}\,\varphi_x & \mathrm{c}\,\varphi_y & \mathrm{s}\,\varphi_x\,\mathrm{s}\,\varphi_y\mathrm{c}^2\,\alpha + \mathrm{s}\,\varphi_x\,\mathrm{s}\,\varphi_y\mathrm{s}^2\,\alpha \end{pmatrix} \tag{D.5}$$

Appendix E

Cubic Splines

Cubic splines are defined by:

$$s(t) = \begin{cases} s_0(t) & \text{for} \quad t \in [t_0, t_1) \\ s_1(t) & \text{for} \quad t \in [t_1, t_1) \\ \dots & \\ s_{n-2}(t) & \text{for} \quad t \in [t_{n-2}, t_{n-1}] \end{cases} \tag{E.1}$$

For each interval, s_i is defined by:

$$s_i(t) = a_i(t - t_i)^3 + b_i(t - t_i)^2 + c_i(t - t_i) + d_i \quad \forall t \in [t_i, t_{i+1}] \tag{E.2}$$

Calculating Coefficients

The spline passes through the control points p_i at $t = t_i$ and is $\mathcal{C}^2$-smooth:

$$s(t_i) = p_i \qquad \forall i \in \{0, \dots, n-1\} \tag{E.3}$$
$$s_i(t_{i+1}) = s_{i+1}(t_{i+1}) \qquad \forall i \in \{0, \dots, n-3\} \tag{E.4}$$
$$\dot{s}_i(t_{i+1}) = \dot{s}_{i+1}(t_{i+1}) \qquad \forall i \in \{0, \dots, n-3\} \tag{E.5}$$
$$\ddot{s}_i(t_{i+1}) = \ddot{s}_{i+1}(t_{i+1}) \qquad \forall i \in \{0, \dots, n-3\} \tag{E.6}$$

These conditions provide $4n - 6$ equations for the $4n - 4$ unknown spline parameters. A unique solution is obtained by setting $\ddot{s}_0(t_0) = \ddot{s}_{n-2}(t_{n-1}) = 0$, resulting in a so-called natural spline. To approximate functions with non-zero initial and/or terminal acceleration, "virtual" control points are added before and/or after the time interval $[t_0, t_{n-1}]$.

Denoting the distance between two control points as $h_i := t_{i+1} - t_i$, we obtain the following equations from the continuity conditions for s and $\ddot{s}$:

$$\begin{aligned} a_i &= \frac{1}{6h_i}(\ddot{p}_{i+1} - \ddot{p}_i) \\ b_i &= \frac{1}{2}\ddot{p}_i \\ c_i &= \frac{1}{h_i}(p_{i+1} - p_i) - \frac{h_i}{6}(\ddot{p}_{i+1} + 2\ddot{p}_i) \\ d_i &= p_i \end{aligned} \tag{E.7}$$

Continuity of $\dot{s}$ finally leads to the following equations for the $n - 2$ unknowns $\ddot{p}_i, i = 1 \dots n-2$:

$$\begin{aligned} & h_{i+1}\ddot{p}_{i-1} + 2(h_{i-1} + h_i)\ddot{p}_i + h_i\ddot{p}_{i+1} = -\frac{6}{h_i}(p_{i+1} - p_i) + \frac{6}{h_{i-1}}(p_i - p_{i-1}) \\ \Leftrightarrow & \boldsymbol{A}\ddot{\boldsymbol{p}} = \boldsymbol{r} \end{aligned} \tag{E.8}$$

Since $\boldsymbol{A}$ is tridiagonal, $\ddot{\boldsymbol{p}}$ can be calculated very efficiently using LR-decomposition. The coefficients are then calculated using (E.7).

Calculating Gradients

Gradients of cubic splines with respect to the parameters $\boldsymbol{p}$ can be calculated using a similar approach to that used for calculating coefficients. Differentiating (E.2) yields:

$$\nabla_{\boldsymbol{p}} \boldsymbol{s} = (\nabla_{\ddot{\boldsymbol{p}}} \boldsymbol{a})\,(\nabla_{\boldsymbol{p}} \ddot{\boldsymbol{p}})\,\boldsymbol{\Delta T}^3 + \nabla_{\ddot{\boldsymbol{p}}} \boldsymbol{b}\,(\nabla_{\boldsymbol{p}} \ddot{\boldsymbol{p}})\,\boldsymbol{\Delta T}^2 + ((\nabla_{\ddot{\boldsymbol{p}}} \boldsymbol{c})\,(\nabla_{\boldsymbol{p}} \ddot{\boldsymbol{p}}) + (\nabla_{\boldsymbol{p}} \boldsymbol{c}))\,\boldsymbol{\Delta T} + (\nabla_{\boldsymbol{p}} \boldsymbol{d}) \quad \text{(E.9)}$$

Here $\boldsymbol{\Delta T}$ is defined as $\mathrm{diag}(t - t_i)$. Gradients of $\dot{s}$ and $\ddot{s}$ are calculated by differentiating (E.9). Differentiating (E.7) yields the gradients of spline coefficients:

$$\frac{\partial a_i}{\partial \ddot{p}_j} = \begin{cases} -\frac{1}{6h_i} & \text{for} \quad j = i \\ \frac{1}{6h_i} & \text{for} \quad j = i+1 \\ 0 & \text{otherwise} \end{cases} \quad \text{(E.10)}$$

$$\frac{\partial b_i}{\partial \ddot{p}_j} = \begin{cases} \frac{1}{2} & \text{for} \quad j = i \\ 0 & \text{otherwise} \end{cases} \quad \text{(E.11)}$$

$$\frac{\partial c_i}{\partial \ddot{p}_j} = \begin{cases} -\frac{1}{3h_i} & \text{for} \quad j = i \\ -\frac{1}{6h_i} & \text{for} \quad j = i+1 \\ 0 & \text{otherwise} \end{cases} \quad \text{(E.12)}$$

$$\frac{\partial c_i}{\partial p_j} = \begin{cases} -\frac{1}{h_i} & \text{for} \quad j = i \\ \frac{1}{h_i} & \text{for} \quad j = i+1 \\ 0 & \text{otherwise} \end{cases} \quad \text{(E.13)}$$

$$\frac{\partial d_i}{\partial p_j} = \begin{cases} 1 & \text{for} \quad j = i \\ 0 & \text{otherwise} \end{cases} \quad \text{(E.14)}$$

Differentiating (E.8) with respect to $\boldsymbol{p}$ yields:

$$\boldsymbol{A}\,(\nabla_{\boldsymbol{p}} \ddot{\boldsymbol{p}}) = \nabla_{\boldsymbol{p}} \boldsymbol{r} \quad \text{(E.15)}$$

The right hand side is given by:

$$\nabla_{\boldsymbol{p}} \boldsymbol{r} = \begin{pmatrix} -\frac{6}{h_0} & \frac{6}{h_0} + \frac{6}{h_1} & -\frac{6}{h_1} & 0 & 0 & 0 & \dots \\ 0 & -\frac{6}{h_1} & \frac{6}{h_1} + \frac{6}{h_2} & -\frac{6}{h_2} & 0 & 0 & \dots \\ 0 & 0 & -\frac{6}{h_2} & \frac{6}{h_2} + \frac{6}{h_3} & -\frac{6}{h_3} & 0 & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \end{pmatrix} \quad \text{(E.16)}$$

$\nabla_{\boldsymbol{p}} \ddot{\boldsymbol{p}}$ is calculated column-wise using an LR-factorization of $\boldsymbol{A}$.
The spline coefficients are linear in the control points:

$$\boldsymbol{a} = (\nabla_{\boldsymbol{p}} \boldsymbol{a})\,\boldsymbol{p} \quad \text{(E.17)}$$

$$\boldsymbol{b} = (\nabla_{\boldsymbol{p}} \boldsymbol{b})\,\boldsymbol{p} \quad \text{(E.18)}$$

$$\boldsymbol{c} = (\nabla_{\boldsymbol{p}} \boldsymbol{c})\,\boldsymbol{p} \quad \text{(E.19)}$$

$$\boldsymbol{d} = (\nabla_{\boldsymbol{p}} \boldsymbol{d})\,\boldsymbol{p} \quad \text{(E.20)}$$

Obviously, $\nabla_{\boldsymbol{p}} \boldsymbol{d}$ is the unit matrix.

Finally, the cubic spline can be written as:

$$\boldsymbol{s}(t) = \left(\nabla_{\boldsymbol{p}}\boldsymbol{a}(t-t_i)^3 + \nabla_{\boldsymbol{p}}\boldsymbol{b}(t-t_i)^2 + \nabla_{\boldsymbol{p}}\boldsymbol{c}(t-t_i) + \nabla_{\boldsymbol{p}}\boldsymbol{d}\right)\boldsymbol{p} = \nabla_{\boldsymbol{p}}\boldsymbol{s}(t)\boldsymbol{p} \tag{E.21}$$

Appendix F

Local Optimization of Kinematic Redundancy

In this thesis, the problem of local optimization of kinematic redundancy is given by:

$$\phi = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}} + \alpha_N \left(\nabla_q H\right) \dot{\boldsymbol{q}} \tag{F.1}$$

$$\boldsymbol{g} = \dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}} \tag{F.2}$$

$$\phi \rightarrow \min! \quad \wedge \quad \boldsymbol{g} = 0 \tag{F.3}$$

Here $\boldsymbol{W}$ is a (usually diagonal) weighting matrix, α_N is a gain factor and H is an auxiliary function. Using the method of Lagrange multipliers, we obtain the Lagrange function:

$$L = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}} + \alpha_N \left(\nabla_q H\right) \dot{\boldsymbol{q}} + \boldsymbol{\lambda}^T (\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}}) \tag{F.4}$$

This leads to the following optimality conditions:

$$\frac{\partial L}{\partial \dot{\boldsymbol{q}}}^T = \boldsymbol{W}\dot{\boldsymbol{q}} + \alpha_N \left(\nabla_q H\right)^T - \boldsymbol{J}^T \boldsymbol{\lambda} \tag{F.5}$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}}^T = \dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}} \tag{F.6}$$

From (F.5) we have:

$$\dot{\boldsymbol{q}} = \boldsymbol{W}^{-1}\boldsymbol{J}^T\boldsymbol{\lambda} - \underbrace{\boldsymbol{W}^{-1}\alpha_N \nabla_q H^T}_{-\boldsymbol{z}} \tag{F.7}$$

Substituting $\dot{\boldsymbol{q}}$ into (F.6), we obtain:

$$\dot{\boldsymbol{x}} = \underbrace{\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T}_{\boldsymbol{B}}\boldsymbol{\lambda} + \boldsymbol{J}\boldsymbol{z}. \tag{F.8}$$

This equation can be solved for $\boldsymbol{\lambda}$, thereby obtaining the closed form solution for $\dot{\boldsymbol{q}}$

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{J}_{\boldsymbol{W}}^{\#}\dot{\boldsymbol{x}} + \boldsymbol{N}_J \boldsymbol{z} \\ &= \boldsymbol{J}_{\boldsymbol{W}}^{\#}\dot{\boldsymbol{x}} - \alpha_N \boldsymbol{W}^{-1}\boldsymbol{N}_J \left(\nabla_q H\right)^T \end{aligned} \tag{F.9}$$

as a function of the $\boldsymbol{W}$-weighted generalized inverse $\boldsymbol{J}_{\boldsymbol{W}}^{\#}$ and the corresponding null-space projection matrix $\boldsymbol{N}$:

$$\boldsymbol{J}_{\boldsymbol{W}}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{J}^T \left(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T\right)^{-1} \tag{F.10}$$

$$\boldsymbol{N}_J = \boldsymbol{E} - \boldsymbol{J}_{\boldsymbol{W}}^{\#}\boldsymbol{J} \tag{F.11}$$

In fact, this closed form solution is often used in controller implementations. However, it is much cheaper to calculate $\dot{\boldsymbol{q}}$ directly by solving (F.8) for $\boldsymbol{\lambda}$ numerically and substituting this into (F.7). This has been suggested by Klein and Huang [71] without a weighting matrix $\boldsymbol{W}$.

Summing up, the general solution for $\dot{\boldsymbol{q}}$ is calculated using the following algorithm:

1: set $\boldsymbol{B} = \boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T$
2: set $\boldsymbol{p} = \dot{\boldsymbol{x}} - \boldsymbol{J}\boldsymbol{z}$
3: solve $\boldsymbol{B}\boldsymbol{\lambda} = \boldsymbol{p}$ for $\boldsymbol{\lambda}$
4: set $\dot{\boldsymbol{q}} = \boldsymbol{W}^{-1}\left(\boldsymbol{J}^T\boldsymbol{\lambda} - \boldsymbol{z}\right)$

Note that $\boldsymbol{W}$ is usually diagonal, which can also be exploited when calculating $\dot{\boldsymbol{q}}$.

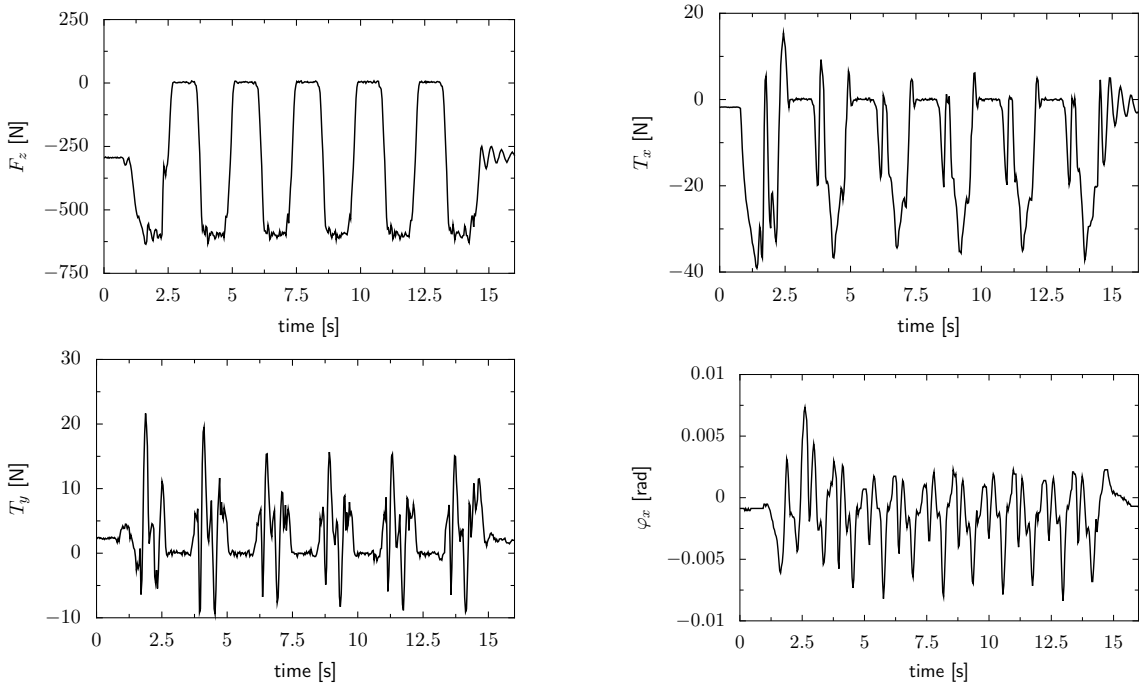
Appendix G

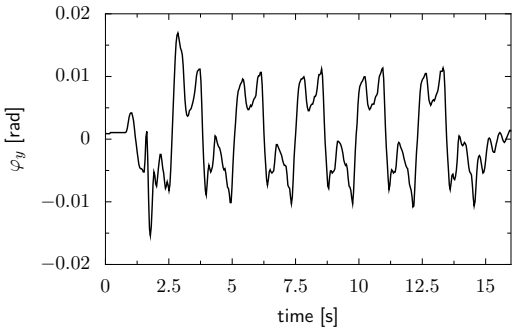
Experimental Results

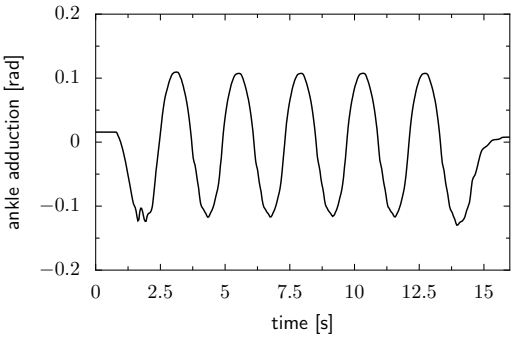
The following sections contain measurements for different walking speeds and directions gathered in experiments with Lola. In all trials, the robot started and stopped in a standing position. Motor-side sensing was used for joint position control and drift compensation was used in the inverse kinematics loop (feedback path 1, see Section 5.4). Only measurements for pelvis and right leg joints are shown for conciseness. The following table lists the basic gait parameters used in the experiments:

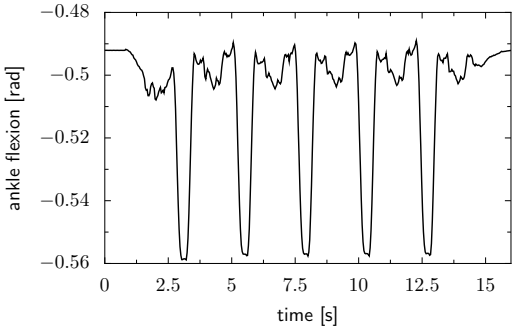
Speed [km/h]	Direction	Step length [cm]	Step duration [s]
0.0	N/A	0	1.2
1.0	forward	34	1.2
2.0	forward	50	0.9
3.0	forward	58	0.7
0.7	sideways	20	1.0

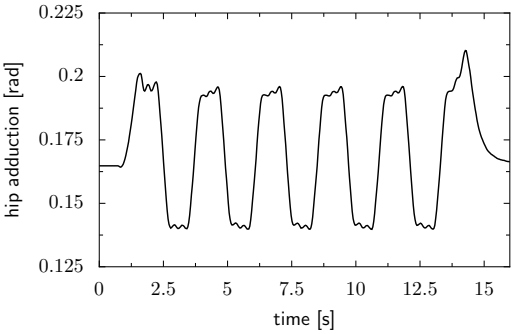
G.1 Walking Forward at 0 km/h

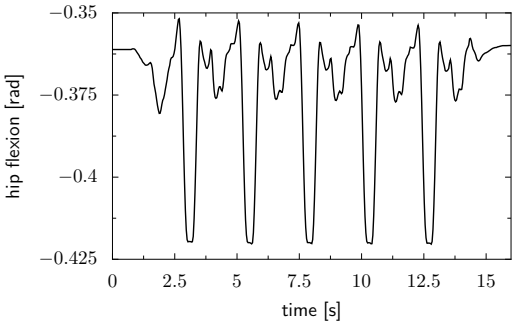


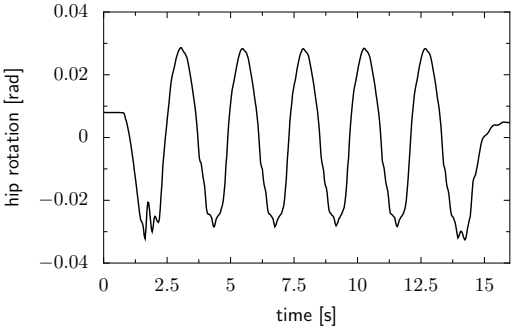

φ_y [rad]
0.02
0.01
0
−0.01
−0.02
0 2.5 5 7.5 10 12.5 15
time [s]

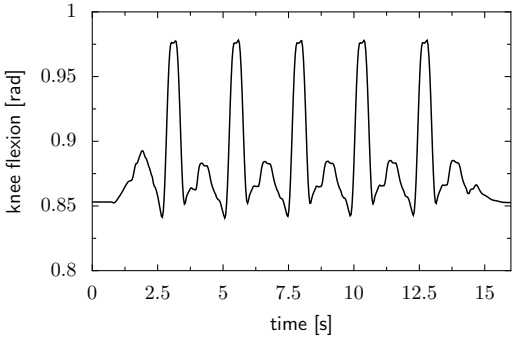

ankle adduction [rad]
0.2
0.1
0
−0.1
−0.2
0 2.5 5 7.5 10 12.5 15
time [s]

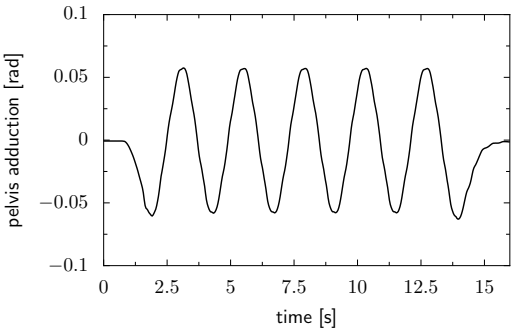

ankle flexion [rad]
−0.48
−0.5
−0.52
−0.54
−0.56
0 2.5 5 7.5 10 12.5 15
time [s]

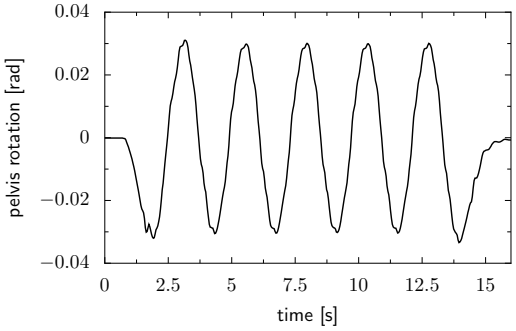

hip adduction [rad]
0.225
0.2
0.175
0.15
0.125
0 2.5 5 7.5 10 12.5 15
time [s]

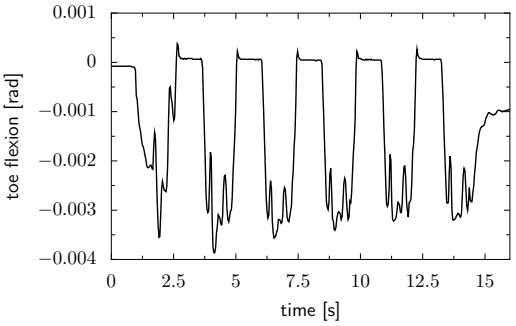

hip flexion [rad]
−0.35
−0.375
−0.4
−0.425
0 2.5 5 7.5 10 12.5 15
time [s]


hip rotation [rad]
0.04
0.02
0
−0.02
−0.04
0 2.5 5 7.5 10 12.5 15
time [s]

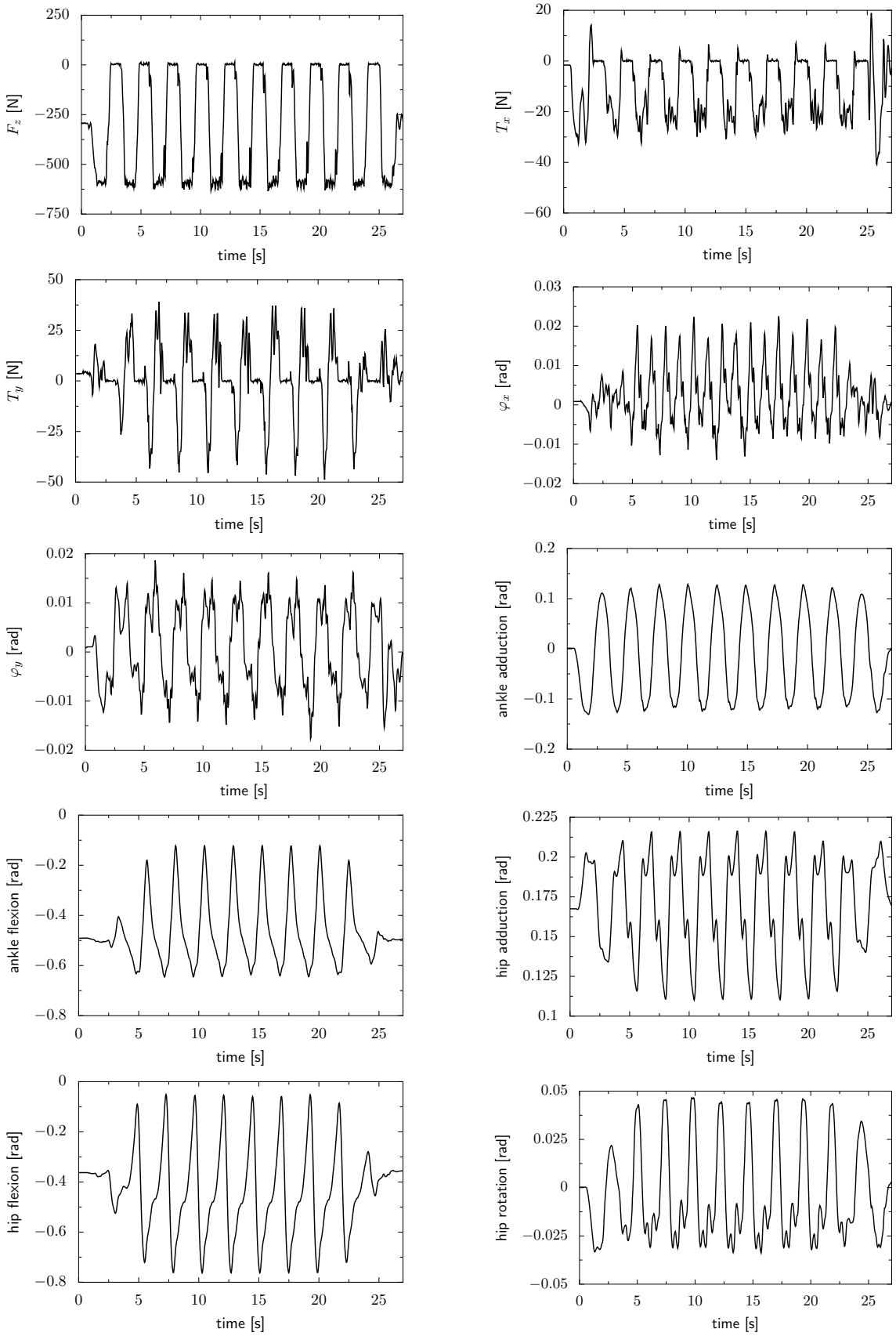

knee flexion [rad]
1
0.95
0.9
0.85
0.8
0 2.5 5 7.5 10 12.5 15
time [s]

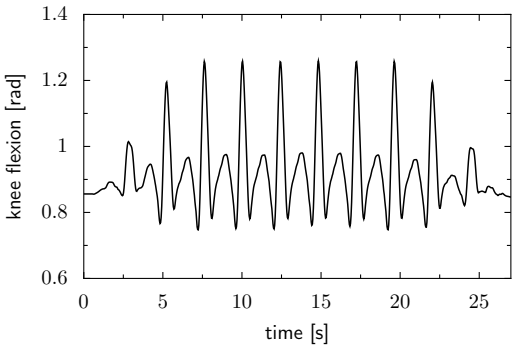

pelvis adduction [rad]
0.1
0.05
0
−0.05
−0.1
0 2.5 5 7.5 10 12.5 15
time [s]

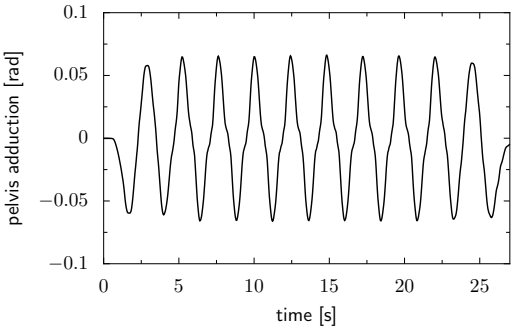

pelvis rotation [rad]
0.04
0.02
0
−0.02
−0.04
0 2.5 5 7.5 10 12.5 15
time [s]

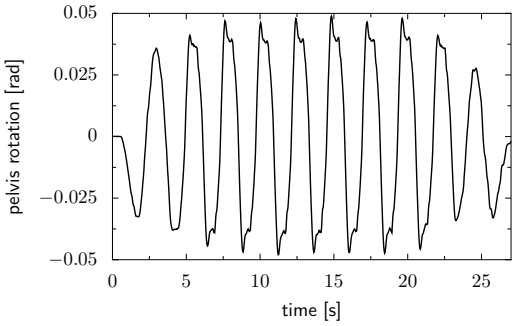

toe flexion [rad]
0.001
0
−0.001
−0.002
−0.003
−0.004
0 2.5 5 7.5 10 12.5 15
time [s]

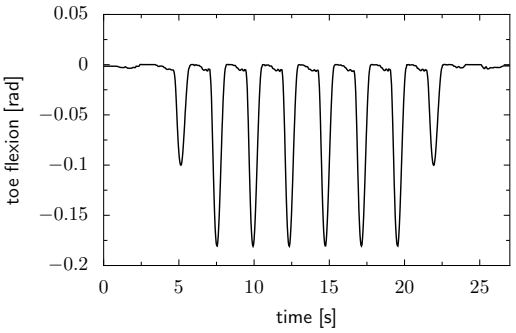
G.2 Walking Forward at 1 km/h



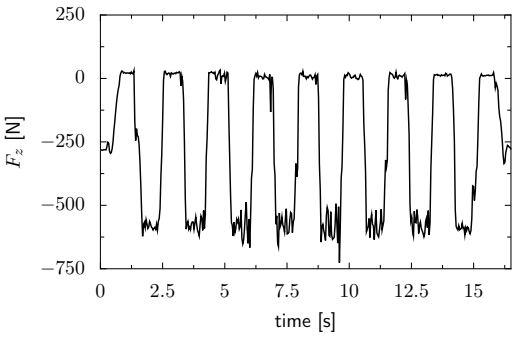


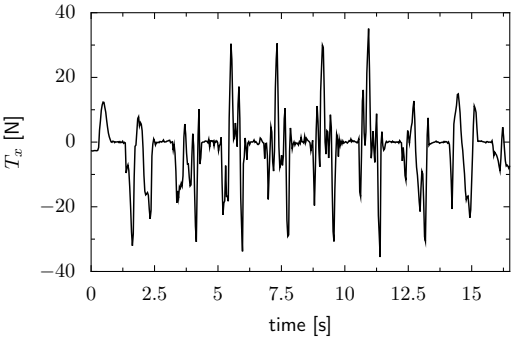


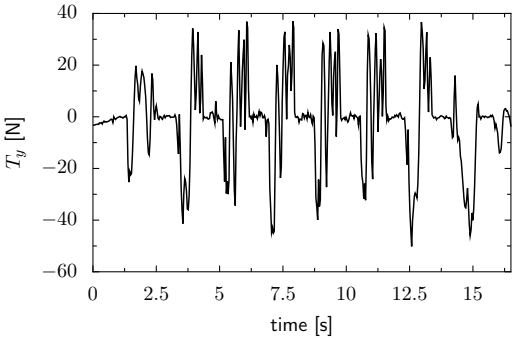


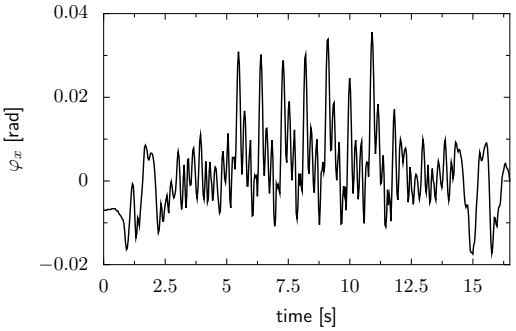


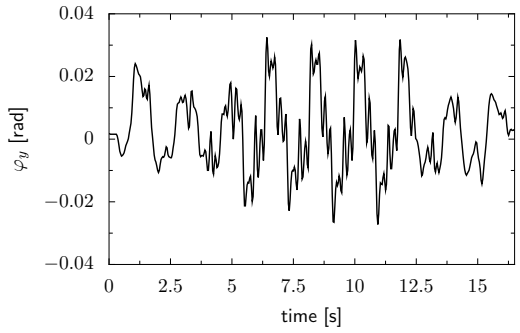
G.3 Walking Forward at 2 km/h

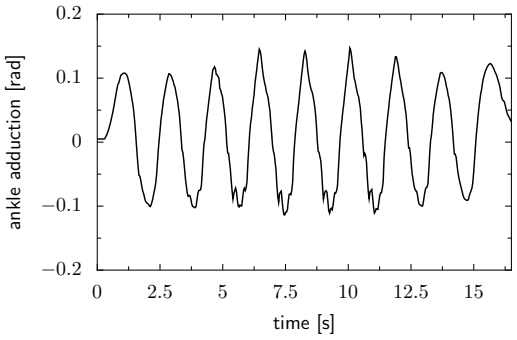


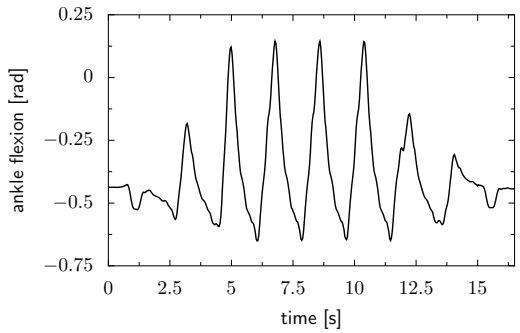


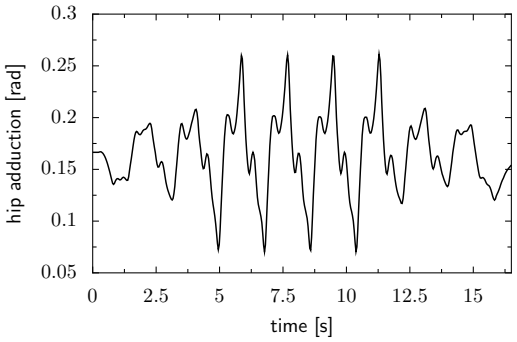


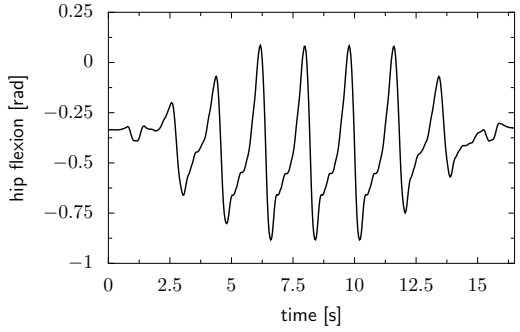


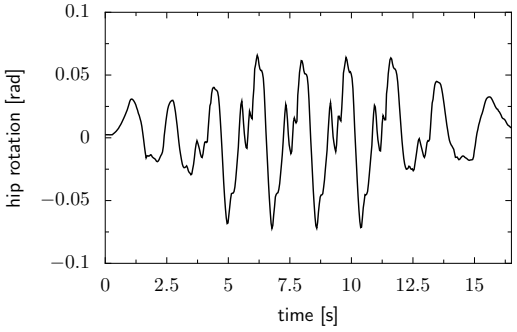

φ_y [rad]
0.04
0.02
0
−0.02
−0.04
0 2.5 5 7.5 10 12.5 15
time [s]

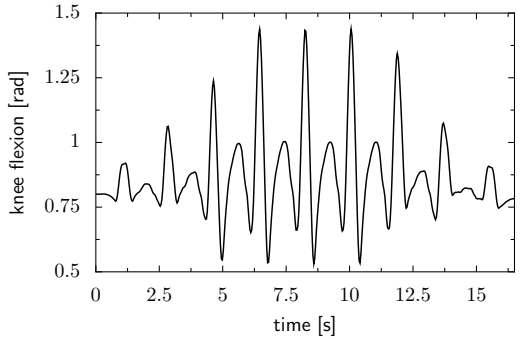

ankle adduction [rad]
0.2
0.1
0
−0.1
−0.2
0 2.5 5 7.5 10 12.5 15
time [s]

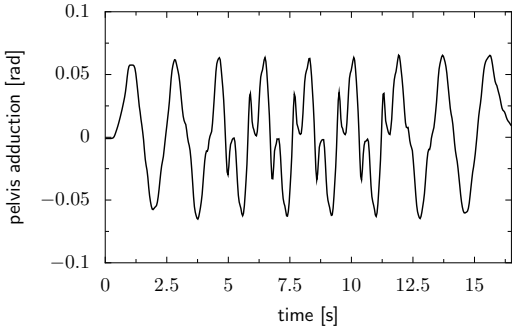

ankle flexion [rad]
0.25
0
−0.25
−0.5
−0.75
0 2.5 5 7.5 10 12.5 15
time [s]

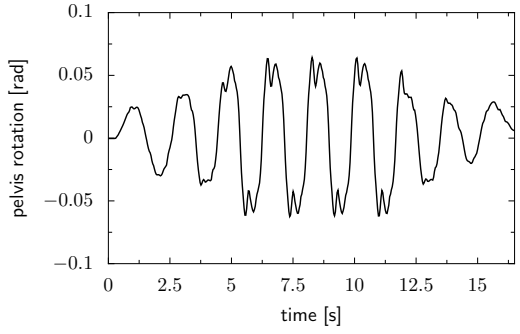

hip adduction [rad]
0.3
0.25
0.2
0.15
0.1
0.05
0 2.5 5 7.5 10 12.5 15
time [s]

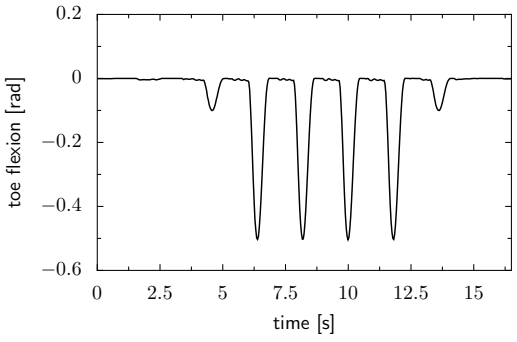

hip flexion [rad]
0.25
0
−0.25
−0.5
−0.75
−1
0 2.5 5 7.5 10 12.5 15
time [s]


hip rotation [rad]
0.1
0.05
0
−0.05
−0.1
0 2.5 5 7.5 10 12.5 15
time [s]

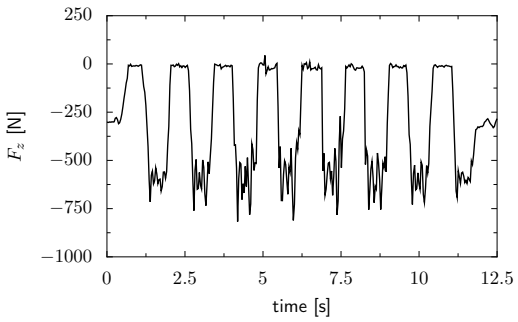

knee flexion [rad]
1.5
1.25
1
0.75
0.5
0 2.5 5 7.5 10 12.5 15
time [s]

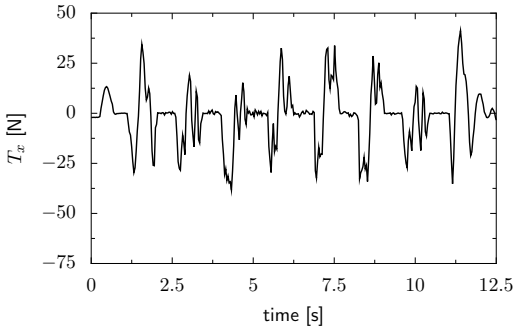

pelvis adduction [rad]
0.1
0.05
0
−0.05
−0.1
0 2.5 5 7.5 10 12.5 15
time [s]

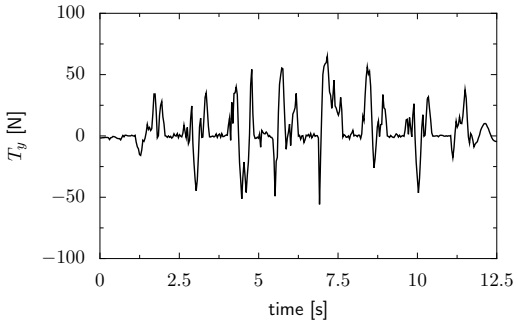

pelvis rotation [rad]
0.1
0.05
0
−0.05
−0.1
0 2.5 5 7.5 10 12.5 15
time [s]

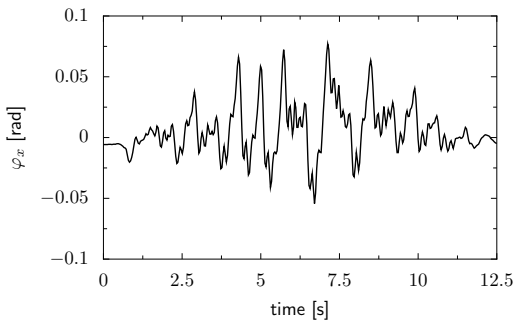

toe flexion [rad]
0.2
0
−0.2
−0.4
−0.6
0 2.5 5 7.5 10 12.5 15
time [s]

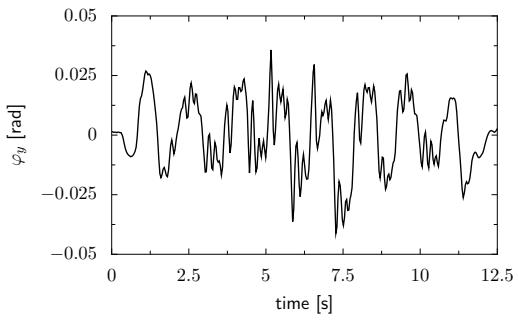
G.4 Walking Forward at 3 km/h

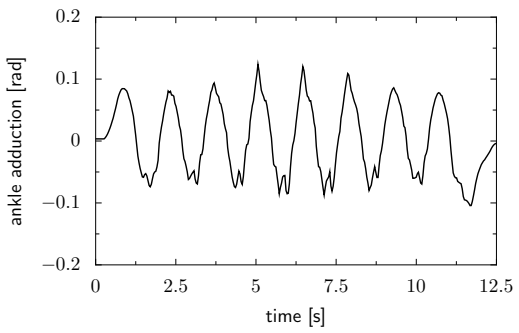


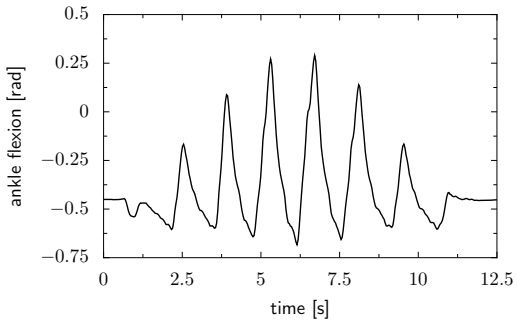


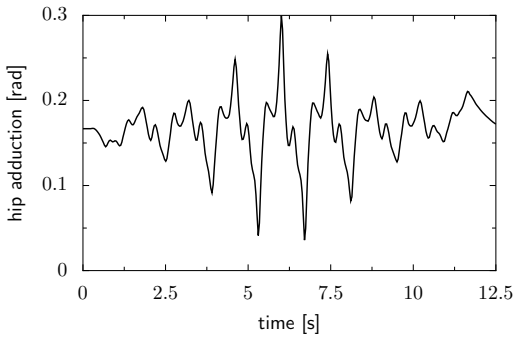


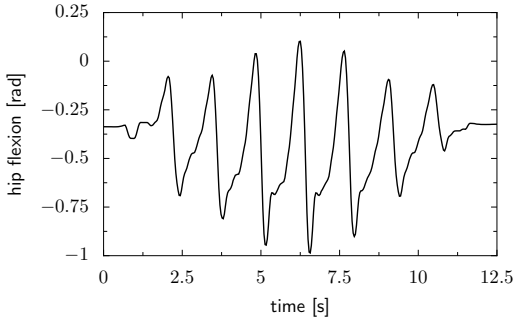


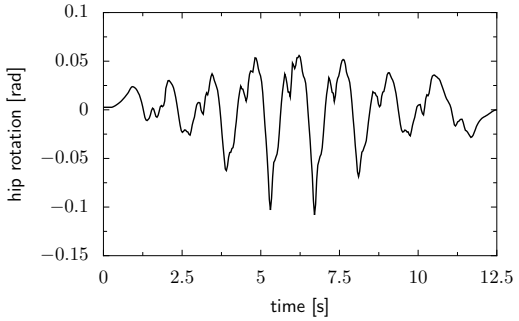


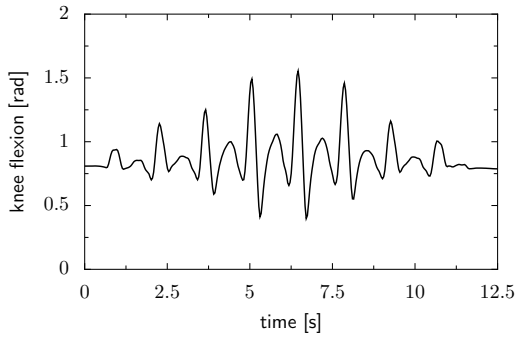


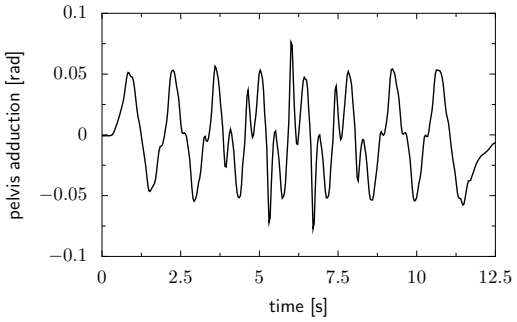


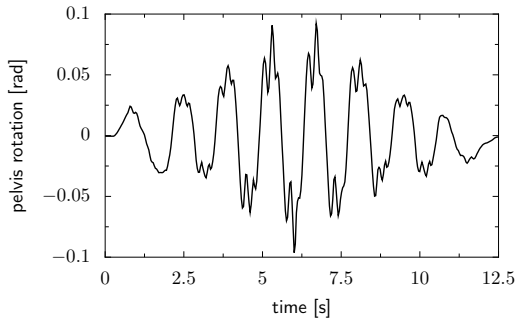


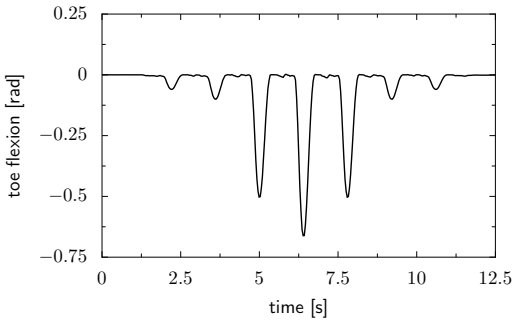




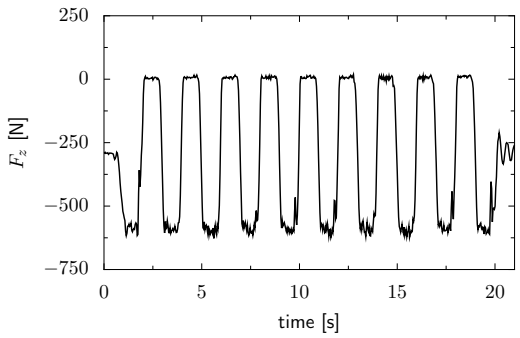


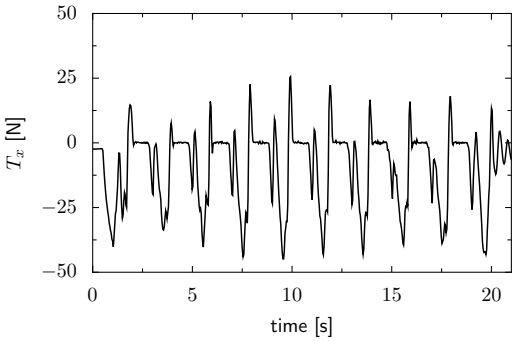


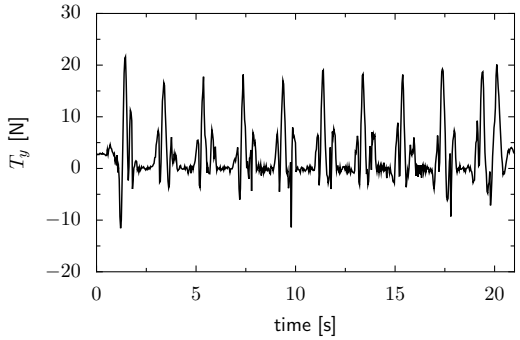


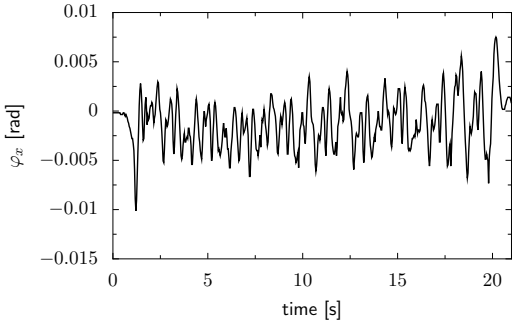


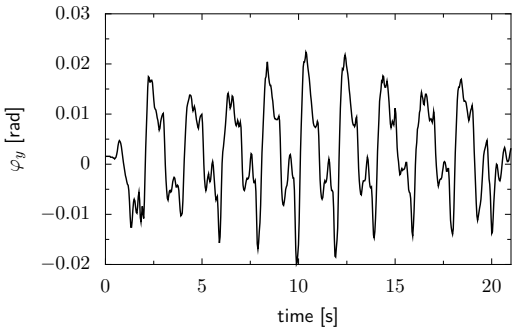
G.5 Walking Sideways at 0.7 km/h

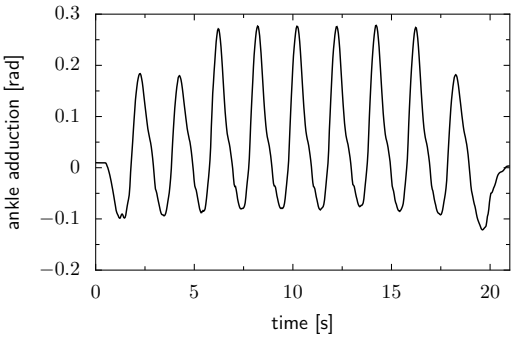


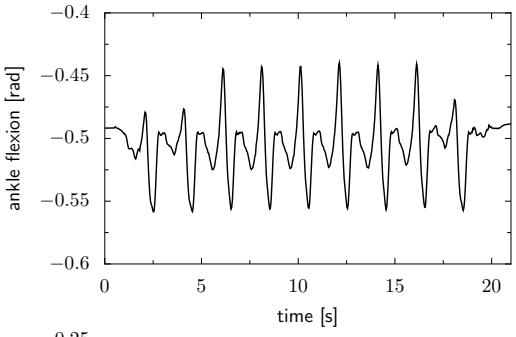


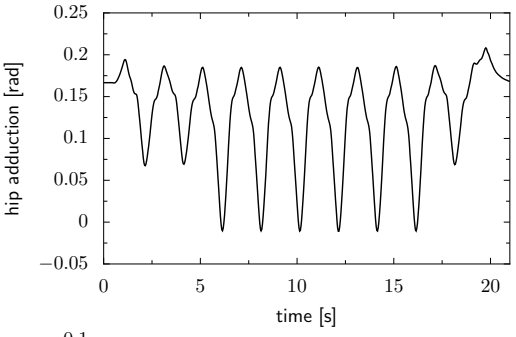


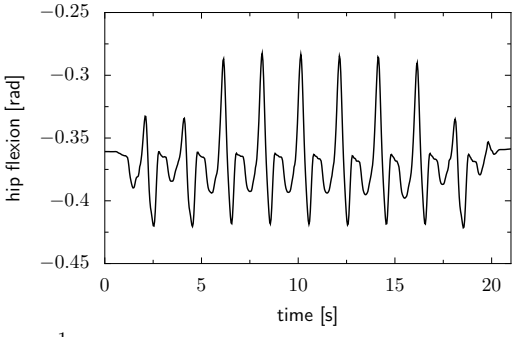


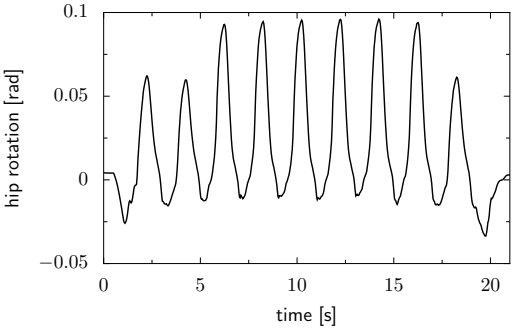

φ_y [rad]
time [s]

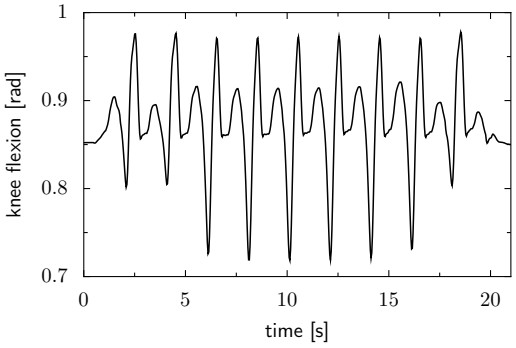

ankle adduction [rad]
time [s]

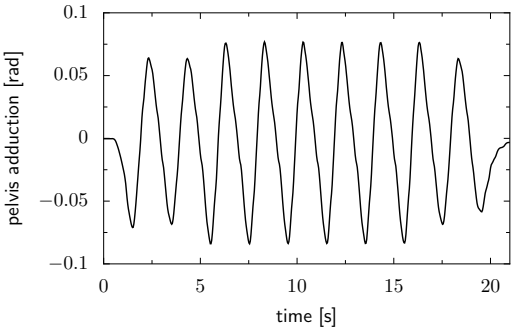

ankle flexion [rad]
time [s]

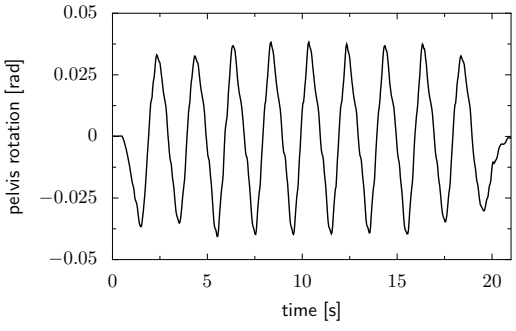

hip adduction [rad]
time [s]

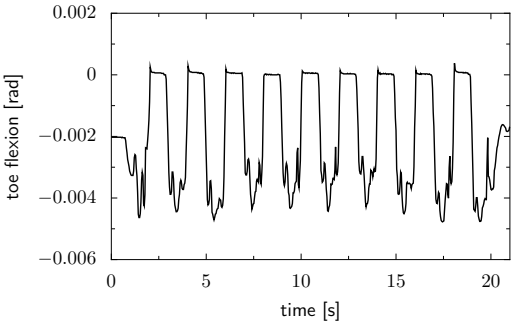

hip flexion [rad]
time [s]


hip rotation [rad]
time [s]


knee flexion [rad]
time [s]


pelvis adduction [rad]
time [s]


pelvis rotation [rad]
time [s]


toe flexion [rad]
time [s]

List of Abbreviations

API	Application Programming Interface
BVP	Boundary Value Problem
CAD	Computer Aided Design
CoG	Center of Gravity
CoP	Center of Pressure
CPU	Central Processing Unit
DoF	Degree of Freedom
EoM	Equation of Motion
FEM	Finite Element Method
FoR	Frame of Reference
FRI	Foot Rotation Indicator
FTS	Force/Torque Sensor
FSW	Feasible Solution of Wrench
GigE	Gigabit Ethernet
GUI	Graphical User Interface
IP	Internet Protocol
IPC	Inter Process Communication
IVP	Initial Value Problem
LIPM	Linear Inverted Pendulum Mode
MBS	Multibody System
ODE	Ordinary Differential Equation
OS	Operating System
PCB	Printed Circuit Board
POSIX	Portable Operating System Interface
PMSM	Permanent Magnet Synchronous Motor
QP	Quadratic Programming
RTOS	Real-Time Operating System
SLAM	Simultaneous Localization and Mapping
SVD	Singular Value Decomposition
TCP	Transmission Control Protocol (a protocol from the Internet Protocol Suite)
TCP	Tool Center Point (usually the reference point of a robot manipulator)
UDP	User Datagram Protocol
ZMP	Zero Moment Point
ZRAM	Zero Rate of Angular Momentum

Bibliography

[1] Acary, V. and Brogliato, B. *Numerical Methods for Nonsmooth Dynamical Systems*. Ed. by Pfeiffer, F. and Wriggers, P. Lecture Notes in Applied and Computational Mechanics. Springer, 2008. DOI: 10.1007/978-3-540-75392-6.

[2] Albu-Schäffer, A. and Hirzinger, G. "Cartesian impedance control techniques for torque controlled light-weight robots". In: vol. 1. 2002, 657 –663 vol.1. DOI: 10.1109/ROBOT.2002.1013433.

[3] Bessonnet, G., Chessé, S., and Sardain, P. "Optimal Gait Synthesis of a Seven-Link Planar Biped". In: *The Intl Journal of Robotics Research* **23**, 10-11 (2004), pp. 1059–1073. DOI: 10.1177/0278364904047393.

[4] BostonDynamics. *Company website for the Petman robot*. 2010/07/25. URL: http://www.bostondynamics.com/robot_petman.html.

[5] Bremer, H. and Pfeiffer, F. *Elastische Mehrkörpersysteme*. (German). Wiesbaden: B.G. Teubner Verlag, 1988.

[6] Buschmann, T., Lohmeier, S., Bachmayer, M., Ulbrich, H., and Pfeiffer, F. "A Collocation Method for Real-Time Walking Pattern Generation". In: *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*. 2007, pp. 1 –6. DOI: 10.1109/ICHR.2007.4813841.

[7] Buschmann, T., Lohmeier, S., Ulbrich, H., and Pfeiffer, F. "Dynamics simulation for a biped robot: modeling and experimental verification". In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. May 2006, pp. 2673 –2678. DOI: 10.1109/ROBOT.2006.1642105.

[8] Buschmann, T., Nishiwaki, K., Inaba, M., and Inoue, H. "Inverse Kinematics for Humanoid Kneeling Motion Exploiting Redundant DOFs". In: *Proc of the Robotics Society of Japan (RSJ)*. 2004.

[9] Buschmann, T., Lohmeier, S., Ulbrich, H., and Pfeiffer, F. "Optimization based gait pattern generation for a biped robot". In: *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*. Dec. 2005, pp. 98 –103. DOI: 10.1109/ICHR.2005.1573552.

[10] Buschmann, T., Lohmeier, S., Schwienbacher, M., Favot, V., Ulbrich, H., Hundelshausen, F. v., Rohe, G., and Wuensche, H.-J. "Walking in Unknown Environments – a Step Towards More Autonomy". In: *Proc IEEE-RAS Int Conference on Humanoid Robotics (Humanoids)*. 2010.

[11] Chareyron, S. and Wieber, P. B. *Stability and regulation of nonsmooth dynamical systems*. Tech. rep. Institut National de Recherche en Informatique et en Automatique (INRIA), 2004.

[12] Chestnutt, J., Kuffner, J., Nishiwaki, K., and Kagami, S. "Planning Biped Navigation Strategies in Complex Environments". In: *Proc IEEE-RAS Int Conference on Humanoid Robotics (Humanoids)*. 2003.

[13] Chevallereau, C., Djoudi, D., and Grizzle, J. "Stable Bipedal Walking With Foot Rotation Through Direct Regulation of the Zero Moment Point". In: *IEEE Trans. Robot. Automat.* **24** (2008), pp. 390–401. DOI: 10.1109/TRO.2007.913563.

[14] Cho, B.-K., Park, S.-S., and Oh, J.-H. "Controllers for running in the humanoid robot, HUBO". In: *Intl Journal of Humanoid Robotics (IJHR)* (Dec. 2009), pp. 385 –390. DOI: 10.1109/ICHR.2009.5379574.

[15] Craig, J. and Raibert, M. "A Systematic Method of Hybrid Position/Force Control of a Manipulator". In: *Computer Software and Applications Conference, IEEE Computer Society*. 1979.

[16] Cupec, R., Lorch, O., and Schmidt, G. "Experiments in Vision-Guided Robot Walking in a Structured Scenario". In: *Proc of the IEEE Intl Symposium on Industrial Electronics*. 2005.

[17] Cupec, R., Lorch, O., and Schmidt, G. "Vision-Guided Humanoid Walking – Concepts and Experiments". In: *Proc. of the 12th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD'03)*. Cassino, Italy, May 2003.

[18] Denk, J. and Schmidt, G. "Synthesis of walking primitive databases for biped robots in 3D-environments". In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*. Vol. 1. Sept. 2003, 1343 –1349 vol.1. DOI: 10.1109/ROBOT.2003.1241778.

[19] Duffy, B. R. "Anthropomorphism and The Social Robot". In: *Robotics and Autonomous Systems* **42** (2003), pp. 177–190. DOI: 10.1016/S0921-8890(02)00374-3.

[20] Engelhardt, T., Friedrich, M., Geier, T., Lebrecht, W., Neumann, L., and Ulbrich, H. "Modelling and optimisation of powertrains". In: *Int. J. Vehicle Systems Modelling and Testing* (2005), pp. 4–31.

[21] Förg, M. *Mehrkörpersysteme mit mengenwertigen Kraftgesetzten – Theorie und Numerik*. Fortschrittberichte VDI, Reihe 20 411. (German). Düsseldorf: VDI-Verlag, 2007. ISBN: 978-3-18-341120-7.

[22] Förg, M., Geier, T., Neumann, L., and Ulbrich, H. "R-Factor Strategies for the Augmented Lagrangian Approach in Multi-Body Contact Mechanics". In: *III European Conference on Computational Mechanics, Solids, Structures and Coupled Problems in Engineering*. 2006.

[23] Fujimoto, Y. and Kawamura, A. "Proposal of biped walking control based on robust hybrid position/force control". In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. Vol. 3. Apr. 1996, 2724 –2730 vol.3. DOI: 10.1109/ROBOT.1996.506574.

[24] Furusho, J. and Masubuchi, M. "A Theoretically Motivated Reduced Order Model for the Control of Dynamic Biped Locomotion". In: *Journal of Dynamic Systems, Measurements, and Control* **109** (1987), pp. 155–163.

[25] Gienger, M. *Entwurf und Realisierung einer zweibeinigen Laufmaschine*. Fortschrittberichte VDI, Reihe 1 378. (German). Düsseldorf: VDI-Verlag, 2005. ISBN: 3-18-337801-9.

[26] Gilmore, P., Kelley, C. T., Miller, C. T., and A., Williams G. "Implicit filtering and optimal design problems: Proceedings of the workshop on optimal design and control". In: *Optimal Design and Control*. 1994.

[27] Goswami, A. "Foot rotation indicator (FRI) point: a new gait planning tool to evaluate postural stability of biped robots". In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 1. 1999, 47 –52 vol.1. DOI: 10.1109/ROBOT.1999.769929.

[28] Goswami, A. and Kallem, V. "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. Apr. 2004, 3785 –3790 Vol.4. DOI: 10.1109/ROBOT.2004.1308858.

[29] Gutmann, J-S., Fukuchi, M., and Fujita, M. "A Floor and Obstacle Height Map for 3D Navigation of a Humanoid Robot". In: *Proc IEEE Int Conf Robotics and Automation (ICRA)*. 2005.

[30] *Harmonic Drive Catalog*. Harmonic Drive AG. Limburg/Lahn, 2007.

[31] Heimsch, Thomas F. and Leine, Remco I. "Lyapunov Stability Theory for Non-Smooth Non-Autonomous Mechanical Systems Applied to the Bouncing Ball Problem". In: *ASME Conference Proceedings* **2009**, 49019 (2009), pp. 465–473. DOI: 10.1115/DETC2009-87185. URL: http://link.aip.org/link/abstract/ASMECP/v2009/i49019/p465/s1.

[32] Hirai, K., Hirose, M., Haikawa, Y., and Takenaka, T. "The development of Honda humanoid robot". In: *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*. Vol. 2. May 1998, 1321 –1326 vol.2. DOI: 10.1109/ROBOT.1998.677288.

[33] Hirukawa, H., Hattori, S., Harada, K., Kajita, S., Kaneko, K., Kanehiro, F., K., Fujiwara., and Morisawa, M. "A Universal Stability Criterion of the Foot Contact of Legged Robots - Adios ZMP-". In: *Proc IEEE Int Conf Robotics and Automation (ICRA)*. 2006.

[34] Hodgins, J.K. and Raibert, M.N. "Adjusting step length for rough terrain locomotion". In: *Robotics and Automation, IEEE Transactions on* **7**, 3 (June 1991), pp. 289 –298. ISSN: 1042-296X. DOI: 10.1109/70.88138.

[35] Honda Motor Company. *History of the Honda robots*. 2010/04/29. URL: http://world.honda.com/ASIMO/history/e4_e5_e6.html.

[36] Huang, Qiang, Kajita, S., Koyachi, N., Kaneko, K., Yokoi, K., Arai, H., Komoriya, K., and Tanie, K. "A high stability, smooth walking pattern for a biped robot". In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 1. 1999, 65 –71 vol.1. DOI: 10.1109/ROBOT.1999.769932.

[37] Hurwitz, A. "Ueber die Bedingungen, unter welchen eine Gleichung nur Wurzeln mit negativen reelen Theilen besitzt." In: *Mathematische Annalen* **46** (1895). (German), pp. 273–284.

[38] Hyon, S. and Emura, T. "Symmetric Walking Control: Invariance and Global Stability". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on.* Apr. 2005, pp. 1443 –1450.

[39] ISO/IEC/IEEE. "Information technology - Portable Operating System Interface (POSIX) Operating System Interface (POSIX)". In: *ISO/IEC/IEEE 9945 (First edition 2009-09-15)* (Sept. 2009), pp. c1 –3830. DOI: 10.1109/IEEESTD.2009.5393893.

[40] Jiménez, P., Thomas, F., and Torras, C. "3D collision detection: a survey". In: *Computers & Graphics* **25** (2001), pp. 269–285.

[41] Kagami, S., Kitagawa, T., Nishiwaki, K., Sugihara, T., Inaba, M., and Inoue, H. "A Fast Dynamically Equilibrated Walking Trajectory Generation Method of Humanoid Robot". In: *Autonomous Robots* **12** (2002), pp. 71–82. DOI: 10.1023/A:1013210909840.

[42] Kagami, S., Nishiwaki, K., Kuffner, J.J., Okada, K., Inaba, M., and Inoue, H. "Vision-based 2.5D terrain modeling for humanoid locomotion". In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on.* Vol. 2. Sept. 2003, 2141 –2146 vol.2.

[43] Kajita, S. and Tani, K. "Experimental study of biped dynamic walking in the linear inverted pendulum mode". In: *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on.* Vol. 3. May 1995, 2885 –2891 vol.3. DOI: 10.1109/ROBOT.1995.525693.

[44] Kajita, S., Nagasaki, T., Kaneko, K., Yokoi, K., and Tanie, K. "A Running Controller of Humanoid Biped HRP-2LR". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on.* Apr. 2005, pp. 616 –622.

[45] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. "Biped walking pattern generation by using preview control of zero-moment point". In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on.* Vol. 2. Sept. 2003, 1620 –1626 vol.2.

[46] Kanehiro, F., Fujiwara, K., Kajita, S., Yokoi, K., Kaneko, K., Hirukawa, H., Nakamura, Y., and Yamane, K. "Open architecture humanoid robotics platform". In: *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on.* Vol. 1. 2002, 24 –30 vol.1. DOI: 10.1109/ROBOT.2002.1013334.

[47] Kaneko, K., Kanehiro, F., Morisawa, M., Miura, K., Nakaoka, S., and Kajita, S. "Cybernetic human HRP-4C". In: *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on.* Dec. 2009, pp. 7 –14. DOI: 10.1109/ICHR.2009.5379537.

[48] Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., and Isozumi, T. "Humanoid robot HRP-2". In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. Vol. 2. Apr. 2004, 1083 –1090 Vol.2. DOI: 10.1109/ROBOT.2004.1307969.

[49] Kaneko, K., Harada, K., Kanehiro, F., Miyamori, G., and Akachi, K. "Humanoid robot HRP-3". In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. Sept. 2008, pp. 2471 –2478. DOI: 10.1109/IROS.2008.4650604.

[50] Karam, W. and Mare, J.-C. "Modelling and simulation of mechanical transmission in roller-screw electromechanical actuators". In: *Aircraft Engineering and Aerospace Technology: An International Journal* **81** (2009), pp. 288–298.

[51] Kawada Industries. *Company website for the HRP-3 robot*. 2010. URL: http://global.kawada.jp/mechatronics/hrp3.html.

[52] Kelly, T. *Implicit Filtering website*. 2007. URL: http://www4.ncsu.edu/~ctk/iffco.html.

[53] Kennedy, C.W. and Desai, J.P. "Modeling and control of the Mitsubishi PA-10 robot arm harmonic drive system". In: *Mechatronics, IEEE/ASME Transactions on* **10**, 3 (June 2005), pp. 263 –274. ISSN: 1083-4435. DOI: 10.1109/TMECH.2005.848290.

[54] Khalil, H. K. *Nonlinear Systems*. 3rd ed. Pearson Education, 2000.

[55] Khatib, O. "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *Robotics and Automation, IEEE Journal of* **3**, 1 (Feb. 1987), pp. 43 –53. ISSN: 0882-4967. DOI: 10.1109/JRA.1987.1087068.

[56] Leigh, R. J. and Zee, D. S. *The Neurology of Eye Movements*. Oxford University Press, Inc., 2006.

[57] Leine, R. I. and Nijmeijer, H. *Dynamics and Bifurcations of Non-smooth Mechanical Systems*. Springer, 2004.

[58] Liégeois, A. "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms". In: *Systems, Man and Cybernetics, IEEE Transactions on* **7**, 12 (Dec. 1977), pp. 868 –871. ISSN: 0018-9472. DOI: 10.1109/TSMC.1977.4309644.

[59] Lin, M. C. and Gottschalk, S. "Collision detection between geometric models: a survey". In: *In Proc. of IMA Conference on Mathematics of Surfaces*. 1998, pp. 37–56.

[60] Löffler, K. *Dynamik und Regelung einer zweibeinigen Laufmaschine*. Fortschrittberichte VDI, Reihe 8 1094. (German). Düsseldorf: VDI-Verlag, 2006. ISBN: 3-18-509408-5.

[61] Löffler, K., Gienger, M., Pfeiffer, F., and Ulbrich, H. "Sensors and control concept of a biped robot". In: *Industrial Electronics, IEEE Transactions on* **51**, 5 (Oct. 2004), pp. 972 –980. ISSN: 0278-0046. DOI: 10.1109/TIE.2004.834948.

[62] Lohmeier, S. "Design and Realization of a Performance Enhanced Humanoid Robot". PhD thesis. Technische Universität München, 2010.

[63] Lyapunov, A. M. *The General Problem of the Stability of Motion*. Taylor & Francis, 1992.

[64] Miura, H. and Shimoyama, I. "Dynamic Walk of a Biped". In: *The International Journal of Robotics Research* **3** (1984), pp. 60–74. DOI: 10.1177/027836498400300206.

[65] Miyazaki, F. and Arimoto, S. "A Control Theoretic Study on Dynamical Biped Locomotion". In: *Journal of Dynamic Systems, Measurement and Control* **102** (1980), pp. 233–239. DOI: 10.1115/1.3149608.

[66] Mombaur, K. D. "Stability Optimization of Open-loop Controlled Walking Robots". PhD thesis. University of Heidelberg, 2001.

[67] Mombaur, K., Bock, H., Schlöder, J., and Longman, R. "Open-loop stable solutions of periodic optimal control problems in robotics". In: *Zeitschrift für Angewandte Mathematik und Mechanik* **7** (2005), pp. 499–515. DOI: 10.1002/zamm.200310190.

[68] Moran, M.E. "Evolution of robotic arms". In: *Journal of Robotic Surgery* **1** (2007), pp. 103–111. DOI: 10.1007/s11701-006-0002-x.

[69] Morisawa, M., Harada, K., Kajita, S., Kaneko, K., Kanehiro, F., Fujiwara, K., Nakaoka, S., and Hirukawa, H. "A Biped Pattern Generation Allowing Immediate Modification of Foot Placement in Real-time". In: *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. Dec. 2006, pp. 581 –586. DOI: 10.1109/ICHR.2006.321332.

[70] Nachtigall, W. and Möhl, B. "Dynamik und Anpassungsvorgänge bei der Laufkoordination des Menschen". In: *Autonomes Laufen*. Ed. by Pfeiffer, F. and Cruse, H. (German). Springer-Verlag Berlin Heidelberg, 2005. Chap. 6, pp. 97–106. ISBN: 978-3-540-25044-9. DOI: 10.1007/3-540-26453-1_6.

[71] Nakamura, Y. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

[72] Nakaoka, S., Hattori, S., Kanehiro, F., Kajita, S., and Hirukawa, H. "Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms". In: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. Oct. 2007, pp. 3641 –3647. DOI: 10.1109/IROS.2007.4399415.

[73] Nishiwaki, K. "Design of walking system and online control of a humanoid robot." in Japanese. PhD thesis. University of Tokyo, 2001.

[74] Nishiwaki, K. and Kagami, S. "High frequency walking pattern generation based on preview control of ZMP". In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. May 2006, pp. 2667 –2672. DOI: 10.1109/ROBOT.2006.1642104.

[75] Nishiwaki, K., Kagami, S., Kuffner, J., Inaba, M., and Inoue, H. "Humanoid "JSK-H7": Research Platform for Autonomous Behavior and Whole Body Motion". In: *Proc Int Workshop Humanoid and human friendly Robotics (IARP)*. Tsukuba, Japan, 2002, pp. 2–9.

[76] Nishiwaki, K., Kagami, S., Kuffner, J.J., Inaba, M., and Inoue, H. "Online humanoid walking control system and a moving goal tracking experiment". In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*. Vol. 1. Sept. 2003, 911 –916 vol.1.

[77] Nishiwaki, K., Kuffner, J., Kagami, S., Inaba, M., and Inoue, H. "The experimental humanoid robot H7: a research platform for autonomous behaviour". In: *Phil. Trans. R. Soc. A* **365**, 1850 (2006), pp. 79–107. DOI: 10.1098/rsta.2006.1921.

[78] Nishiwaki, Koichi, Kuga, Mamoru, Inaba, Satoshi Kagami Masayuki, and inoue, Hirochika. "Whole-body Cooperative Balanced Motion Generation for Reaching". In: *Proc IEEE-RAS Int Conference on Humanoid Robotics (Humanoids)*. 2004.

[79] Ogura, Y., Aikawa, H., Shimomura, K., Morishima, A., Lim, Hun ok, and Takanishi, A. "Development of a new humanoid robot WABIAN-2". In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. May 2006, pp. 76 –81. DOI: 10.1109/ROBOT.2006.1641164.

[80] Oh, J.-H. *Website for the Hubo2 robot*. 2010. URL: http://hubolab.co.kr/hubo%28khr-4%29_Specification.php.

[81] Okada, K., Ogura, T., Haneda, A., and Inaba, M. "Autonomous 3D walking system for a humanoid robot based on visual step recognition and 3D foot step planner". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. Apr. 2005, pp. 623 –628.

[82] Okada, K., Kagami, S., Inaba, M., and Inoue, H. "Plane Segment Finder: Algorithm, Implementation and Applications". In: *Proc IEEE Int Conf Robotics and Automation (ICRA)*. 2001, pp. 2120–5.

[83] Okada, K., Inaba, M., and Inoue, H. "Real-time and Precise Self Collision Detection System for Humanoid Robots". In: *Proc IEEE Int Conf Robotics and Automation (ICRA)*. 2005.

[84] Papadopoulos, E. G. and Chasparis, G. C. "Analysis and model-based control of servomechanisms with friction". In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. Vol. 3. 2002, 2109 –2114 vol.3. DOI: 10.1109/IRDS.2002.1041578.

[85] Park, Jong H. and Kim, Kyoung D. "Biped Robot Walking Using Gravity-Compensated Inverted Pendulum Mode and Computed Torque Control". In: *Proc IEEE Int Conf Robotics and Automation (ICRA)*. 1998.

[86] Parks, P. C. "A. M. Lyapunov's stability theory – 100 years on". In: *IMA Journal of Mathematical Control & Information* **9** (1992), pp. 275–3.

[87] Peer, A., Bajcinca, N., and Schweiger, C. "Physical-based Friction Identification of an Electro-Mechanical Actuator with Dymola/Modelica and MOPS". In: *Proc. of the 3rd Intl. Modelica Conf.* Linköping, 2003.

[88] Perry, J. *Gait Analysis – Normal and Pathological Function*. 3rd edition. Thorofare, USA: Slack Inc., 1992.

[89] Pfeiffer, F. "The TUM walking machines". In: *Phil. Trans. R. Soc. A* **365** (2006), pp. 109–131. DOI: 10.1098/rsta.2006.1922.

[90] Pfeiffer, F. and Cruse, H., eds. *Autonomes Laufen*. (German). Springer-Verlag Berlin Heidelberg, 2005. ISBN: 978-3-540-25044-9. DOI: 10.1007/b137612.

[91] Pfeiffer, F. and Glocker, Ch. *Multibody Dynamics with Unilateral Contacts*. New York: John Wiley Inc., 1996.

[92] Pfeiffer, F. and Johanni, R. "A Concept for Manipulator Trajectory Planning". In: *Proc IEEE Int Conf Robotics and Automation (ICRA)*. 1986. DOI: 10.1109/JRA.1987.1087090.

[93] Pratt, J. E., Krupp, B., Ragusila, V., Rebula, J., et al. "The Yobotics-IHMC Lower Body Humanoid Robot". In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. Oct. 2009, pp. 410 –411. DOI: 10.1109/IROS.2009.5354430.

[94] Pratt, J. *Website for the M2 robot*. 2001. URL: http://www.ai.mit.edu/projects/leglab/robots/m2/m2.html.

[95] Pratt, J., Carff, J., Drakunov, S., and Goswami, A. "Capture Point: A Step toward Humanoid Push Recovery". In: *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. Genova, Dec. 2006, pp. 200–207. DOI: 10.1109/ICHR.2006.321385.

[96] Radhakrishnan, K. and Hindmarsh, A. C. *Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations*. Tech. rep. Lawrence Livermore National Laboratory Report UCRL-ID-113855, 1993.

[97] Raibert, M. H. *Legged Robots that Balance (Artificial Intelligence)*. Cambridge: MIT Press, 1986.

[98] Raibert, M., Blankespoor, K., Nelson, G., and Playter, R. "BigDog, the Rough-Terrain Quadruped Robot". In: *Proc. of the 17th World Congress The International Federation of Automatic Control*. 2008.

[99] Raibert, Marc. "Legged Robots". In: *Communications of the ACM* **26** (1986), pp. 499–514.

[100] Riebe, S. and Ulbrich, H. "Modelling and Control of a Low-Frequency Active Vibration Isolation with Six Degrees-of-Freedom". In: *Proc. of the EUROMECH Colloq. 455 on Semi-Active Vibration Suppression.* Prag, Tschechien, July 2005, pp. 1–15.

[101] Robinson, D., Pratt, J., Paluska, D., and Pratt, G. "Series Elastic Actuator Development for a Biomimetic Walking Robot". In: *Proc. of the ASME International Conference on Advanced Intelligent Mechatronics.* Atlanta, Georgia, USA, 1999, pp. 561–568.

[102] Rockafellar, R. T. "Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming". In: *MATHEMATICS OF OPERATIONS RESEARCH* **1**, 2 (1976), pp. 97–116. DOI: 10.1287/moor.1.2.97.

[103] Rossmann, Th. *Eine Laufmaschine für Rohre.* Fortschrittberichte VDI, Reihe 8 Nr. 732. (German). Düsseldorf: VDI-Verlag, 1998.

[104] Russel, R.D. and Shampine, L.F. "A Collocation Method for Boundary Value Problems". In: *Numer. Math., Springer Verlag* **19** (1972), pp. 1–28.

[105] Sabe, K., Fukuchi, M., Gutmann, J.-S., Ohashi, T., Kawamoto, K., and Yoshigahara, T. "Obstacle avoidance and path planning for humanoid robots using stereo vision". In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on.* Vol. 1. Apr. 2004, 592 –597 Vol.1. DOI: 10.1109/ROBOT.2004.1307213.

[106] Sardain, P. and Bessonnet, G. "Forces acting on a biped robot. Center of pressure-zero moment point". In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* **34**, 5 (Sept. 2004), pp. 630 –637. ISSN: 1083-4427. DOI: 10.1109/TSMCA.2004.832811.

[107] Schiehlen, W., Guse, N., and Seifried, R. "Multibody dynamics in computational mechanics and engineering applications". In: *Computer Methods in Applied Mechanics and Engineering* **195**, 41-43 (2006). John H. Argyris Memorial Issue. Part II, pp. 5509 –5522. ISSN: 0045-7825. DOI: DOI: 10.1016/j.cma.2005.04.024.

[108] Schröder, D. *Elektrische Antriebe – Regelung von Antriebssystemen.* 3rd ed. (German). Springer-Verlag Berlin Heidelberg, 2001.

[109] Shevitz, D. and Paden, B. "Lyapunov stability theory of nonsmooth systems". In: *Automatic Control, IEEE Transactions on* **39**, 9 (Sept. 1994), pp. 1910 –1914. ISSN: 0018-9286. DOI: 10.1109/9.317122.

[110] Siciliano, B. and Khatib, O., eds. *Springer Handbook of Robotics.* Springer-Verlag Berlin Heidelberg, 2008.

[111] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. *Robotics.* Ed. by Grimble, J. and Johnson, A. M. Springer Verlag London Ltd., 2009. DOI: 10.1007/978-1-84628-642-1.

[112] Spong, M.W. "Partial feedback linearization of underactuated mechanical systems". In: *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*. Vol. 1. Sept. 1994, 314 –321 vol.1. DOI: 10.1109/IROS.1994.407375.

[113] Stovall, S. H. *Basic Inertial Navigation*. Tech. rep. Naval Air Warfare Center Weapons Division, 1997.

[114] Taghirad, H. D. and Bélanger, P. R. "Modelling and Parameter Identification of Harmonic Drive Systems". In: *Journal of Dynamic Systems, Measurements, and Control* **120** (1998), pp. 439–444. DOI: doi:10.1115/1.2801484.

[115] Taghirad, H. D. and Bélanger, P.R. "A Nonlinear Model For Harmonic Drive Friction and Compliance". In: *Intl. Conf on Robotics and Automation*. 1998.

[116] Tajima, R., Honda, D., and Suga, K. "Fast running experiments involving a humanoid robot". In: *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*. May 2009, pp. 1571 –1576. DOI: 10.1109/ROBOT.2009.5152404.

[117] Takanishi, A. *Website of the Biped Humanoid Robot Group at Waseda University*. 2010/07/03. URL: http://www.takanishi.mech.waseda.ac.jp/top/research/index.htm.

[118] Takao, S., Yokokohji, Y., and Yoshikawa, T. "FSW (feasible solution of wrench) for multi-legged robots". In: *Proc. IEEE Int. Conf. Robotics and Automation ICRA '03*. Vol. 3. 2003, pp. 3815–3820. DOI: 10.1109/ROBOT.2003.1242182.

[119] Takenaka, T. *Controller of Legged Mobile Robot*. European Patent Application no. EP1475198A1. 2004.

[120] Takenaka, T. *Gait generation system of legged mobile robot.* european patent application no. EP1671754A2. 2004.

[121] Takenaka, T., Matsumoto, T., and Yoshiike, T. "Real time motion generation and control for biped robot -1st report: Walking gait pattern generation-". In: *Proc IEEE/RSJ Int Conf Intelligent Robots and Systems (IROS)*. Oct. 2009, pp. 1084 –1091. DOI: 10.1109/IROS.2009.5354662.

[122] Takenaka, T., Matsumoto, T., Yoshiike, T., and Shirokura, S. "Real time motion generation and control for biped robot -2nd report: Running gait pattern generation-". In: *Proc IEEE/RSJ Int Conf Intelligent Robots and Systems (IROS)*. Oct. 2009, pp. 1092 –1099. DOI: 10.1109/IROS.2009.5354654.

[123] Takenaka, T., Matsumoto, T., and Yoshiike, T. "Real time motion generation and control for biped robot -3rd report: Dynamics error compensation-". In: *Proc IEEE/RSJ Int Conf Intelligent Robots and Systems (IROS)*. Oct. 2009, pp. 1594 –1600. DOI: 10.1109/IROS.2009.5354542.

[124] Takenaka, T., Matsumoto, T., Yoshiike, T., Hasegawa, T., Shirokura, S., Kaneko, H., and Orita, A. "Real time motion generation and control for biped robot -4th report: Integrated balance control-". In: *Proc IEEE/RSJ Int Conf Intelligent Robots and Systems (IROS)*. Oct. 2009, pp. 1601 –1608. DOI: 10.1109/IROS.2009.5354522.

[125] Toyota Motor Corporation. *Website of the Toyota Motor Corporation, Overview of the partner robots*. 2008/05/03. URL: http://www.toyota.co.jp/en/special/robot/.

[126] Ulbrich, H. *Maschinendynamik*. (German). Wiesbaden: B.G. Teubner Verlag, 1996.

[127] von Hundelshausen, F., Himmelsbach, M., Hecker, F., Mueller, A., and Wuensche, H.-J. "Driving with tentacles: Integral structures for sensing and motion". In: *J. Field Robot.* **25**, 9 (2008), pp. 640–73. ISSN: 1556-4959. DOI: 10.1002/rob.v25:9.

[128] Vukobratović, J., Borovac, B., Surla, D., and Stokic, D. *Biped Locomotion: Dynamics, Stability, Control and Applications*. Berlin, Heidelberg, New York: Springer, 1990.

[129] Vukobratović, M. and Borovac, B. "Zero-Moment Point - Thirty Five Years of its Life". In: *Intl Journal of Humanoid Robotics (IJHR)* **1**, 1 (2004), pp. 157–173. DOI: 10.1142/S0219843604000083.

[130] Whitney, D. E. "Resolved Motion Rate Control of Manipulators and Human Prostheses". In: *IEEE Transactions on Man Machine Systems* **10**, 2 (June 1969), pp. 47–53. ISSN: 0536-1540. DOI: 10.1109/TMMS.1969.299896.

[131] Wieber, P. B. "On the stability of walking systems". In: *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*. 2002.

[132] Wieber, P. B., Billet, F., Boissieux, L., and Pissard-Gibollet, R. "The HuMAnS toolbox, a homogeneous framework for motion capture, analysis and simulation". In: *Proc. 9th ISB Symposium on 3D analysis of human movement*. 2006.

[133] Wriggers, P. *Computational Contact Mechanics*. John Wiley & Sons Ltd., 2002.

[134] Yamane, Katsu. *Simulating and Generating Motions of Human Figures*. Springer-Verlag Berlin Heidelberg, 2004.

[135] Zubov, V.I. *Methods of A.M. Lyapunov and their Application*. Ed. by Boron, Leo F. In AM-Bib. Noordhoff, 1964.