ANALYTIC AND LEARNED FOOTSTEP CONTROL FOR ROBUST BIPEDAL WALKING

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

MARCELL MISSURA

aus

Eger, Ungarn

Bonn, März 2015

Angefertigt mit Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Sven Behnke
2. Gutachter: Prof. Dr. Maren Bennewitz

Tag der Promotion: 12.02.2016
Erscheinungsjahr: 2016

Mamának

*31.03.1958 – †16.03.2009

ABSTRACT

Bipedal walking is a complex, balance-critical whole-body motion with inherently unstable inverted pendulum-like dynamics. Strong disturbances must be quickly responded to by altering the walking motion and placing the next step in the right place at the right time. Unfortunately, the high number of degrees of freedom of the humanoid body makes the fast computation of well-placed steps a particularly challenging task. Sensor noise, imprecise actuation, and latency in the sensomotoric feedback loop impose further challenges when controlling real hardware.

This dissertation addresses these challenges and describes a method of generating a robust walking motion for bipedal robots. Fast modification of footstep placement and timing allows agile control of the walking velocity and the absorption of strong disturbances. In a divide and conquer manner, the concepts of motion and balance are solved separately from each other, and consolidated in a way that a low-dimensional balance controller controls the timing and the footstep locations of a high-dimensional motion generator. Central pattern generated oscillatory motion signals are used for the synthesis of an open-loop stable walk on flat ground, which lacks the ability to respond to disturbances due to the absence of feedback. The Central Pattern Generator exhibits a low-dimensional parameter set to influence the timing and the landing coordinates of the swing foot.

For balance control, a simple inverted pendulum-based physical model is used to represent the principal dynamics of walking. The model is robust to disturbances in a way that it returns to an ideal trajectory from a wide range of initial conditions by employing a combination of Zero Moment Point control, step timing, and foot placement strategies. The simulation of the model and its controller output are computed efficiently in closed form, supporting high-frequency balance control at the cost of an insignificant computational load. Additionally, the sagittal step size produced by the controller can be trained online during walking with a novel, gradient descent-based machine learning method. While the analytic controller forms the core of reliable walking, the trained sagittal step size complements the analytic controller in order to improve the overall walking performance. The balanced whole-body walking motion arises by using the footstep coordinates and the step timing predicted by the low-dimensional model as control input for the Central Pattern Generator. Real robot experiments are presented as evidence for disturbance-resistant, omnidirectional gait control, with arguably the strongest push-recovery capabilities to date.

ZUSAMMENFASSUNG

Der zweibeinige Gang ist eine komplexe, balancekritische Ganzkörperbewegung mit der inherent instabilen Dynamik eines inversen Pendels. Starke Störungen erfordern eine schnelle Reaktion mit einem Schritt zum richtigen Zeitpunkt an die richtige Stelle. Leider stellt die hohe Anzahl der Freiheitsgrade eines humanoiden Roboters eine besondere Herausforderung für die schnelle Erzeugung von gut platzierten Schritten dar. Sensorrauschen, unpräzise Motorik und Latenz in der sensomotorischen Schleife erschweren im Betrieb auf realer Hardware diese Aufgabe zusätzlich.

In dieser Dissertation wird eine Methode vorgestellt, die eine robuste Gehbewegung für humanoide Roboter erzeugt. Schnelle Modifikation des Timings und der Schrittkoordinaten ermöglichen eine agile Steuerung der Fortbewegungsgeschwindigkeit, sowie die Rückgewinnung der Stabilität nach starken Störungen. Im Sinne einer "Divide and Conquer" Strategie werden die Körperbewegung und die Balanceregelung als getrennte Konzepte gelöst und zusammengeführt, indem ein niedrigdimensionales Balancemodell die Steuerung eines Bewegungsgenerators für Schrittbewegungen mittels Schrittkoordinaten und Timing übernimmt. Für die Synthese einer Gehbewegung werden mit einem zentralen Mustergenerator periodische Bewegungssignale erzeugt, die auch ohne geschlossenen Regelkreis auf ebenem Untergrund stabil sind, aber nicht auf äußere Störungen reagieren können.

Für die Balanceregelung wird ein einfaches physikales Modell mit der Dynamik eines inversen Pendels verwendet. Das Modell reagiert robust auf Störungen indem es durch eine Kombination aus Druckpunktregelung, Timing und Schrittplatzierung aus einer Vielzahl von instabilen Zuständen zu einer idealen Trajektorie zurückkehrt. Die Simulation sowie die Ausgangsgrößen des physikalischen Modells werden dabei effizient in geschlossener Form berechnet. Weiterhin kommt eine neuartige maschinelle Lernmethode zum Einsatz, die mittels einer einfachen Modellannahme einen Gradienten schätzt und die sagittale Schrittgröße online während der Gehbewegung trainiert. Während die analytische Balanceregelung eine stabile Gehbewegung initialisert, kann das Lernverfahren durch eine gelernte Differenz zu der Ausgabegröße der analytischen Regelung die Stabilität der Gehbewegung zusätzlich verbessern. Die balancierte Ganzkörperbewegung entsteht indem die mit dem Punktmassemodell vorhergesagten Schrittkoordinaten und Zeitpunkte als Steuereingaben für den Mustergenerator verwendet werden. Experimente mit realen Robotern belegen, dass die analytisch berechnete und online trainierte Schrittregelung eine störungsresistente Gehbewegung erzeugt.

PUBLICATIONS

The following publications contributed to this thesis:

- Development of a 2D Linear Inverted Pendulum Model-based gait controller that incorporates conceptual determinants of walking in analytic form. The controller computes Zero Moment Point, step timing, and footstep location parameters that return the center of mass from a wide range of initial states to an ideal trajectory for a desired walking pace. The parameters are computed in closed form, which allows for a fast controller response in the sub-millisecond range.

  Missura and Behnke [2011]
  Alcaraz-Jiménez et al. [2012]
  Missura and Behnke [2013b]

- Real hardware evaluation of the analytic balance model in combination with a central pattern generated gait. The step size and step timing outputs of the balance model are transformed into control signals of a central pattern generator to realize targeted and timed steps. The combination of the two technologies produces a robust omnidirectional walk that challenges the state of the art.

  Missura and Behnke [2013a]
  Missura and Behnke [2014b]

- Development of an online machine learning concept that learns a foot placement controller during walking. The learning process addresses the trade-off between stepping onto a desired location and maintaining balance. Using a simple model assumption, a gradient is computed that allows for fast learning from errors measured at the end of each step. The publications include experiments with a simulated robot. Fast learning from the experience of failed steps during falling, and strong push recovery capabilities have been achieved with a real bipedal robot using this method.

  Missura and Behnke [2014a]
  Missura et al. [2014]

A huge gap exists between what we know is possible with today's machines and what we have so far been able to finish.

— Donald Knuth

ACKNOWLEDGMENTS

My deepest gratitude goes to my wife Olana, without whom all of this would not have been possible, and would not make sense. I thank my father, Béla Missura, who taught me the meaning of unconditional support. I thank Hannes Schulz, Jens Behley, and Philipp Allgeuer for insightful discussions, motivation, and for helping out with countless cumbersome little things. I am grateful to Michael Schreiber for building these excellent humanoid robots and actively supporting my research with prompt no questions asked maintenance work that made it possible to go to the limits of these machines and a little bit beyond. I thank Prof. Dr. Sven Behnke for the opportunity to study bipedal walking in his laboratory and for supporting my research with know-how and reliable financial support.

I am also grateful for a number of freely available software tools that are the fruits of voluntary work by people all around the world. Among these are the GNU C++ compiler, the Eclipse IDE, the CVS and SVN version control systems, the Armadillo linear algebra library, the LWPR library for incremental learning, the Qt framework, the libQGLViewer library, OpenGL, OpenOffice, gnuplot, and the Bullet Physics Engine.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF VIDEOS

ACRONYMS

CoM	Center of Mass
CoP	Center of Pressure
CPG	Central Pattern Generator
LIPM	Linear Inverted Pendulum Model
IMU	Inertial Measurement Unit
ZMP	Zero Moment Point

1 INTRODUCTION

Bipedal walking is an energy efficient and versatile means of locomotion that has been used by vertebrates for over 200 million years. The principle of vaulting over a stiff support leg combined with a ballistic motion of the swing leg results in an almost effortless way of transportation, suitable for covering large distances. Moreover, bipeds, in particular humans, are adept in a diversity of terrains. They can walk over slippery, muddy, and rough terrain, leap over gaps, climb steep structures, and swim in water. The key to truly mobile humanoid robots that can move about freely in human environments, walk up stairs, step over obstacles, and use vehicles that were designed to be operated by a pair of legs, lies in the synthesis of the efficiency and robustness of the natural human gait.

Bipedal locomotion excels in energy efficiency and versatility.

Unfortunately, two-legged robots are inherently unstable and rather difficult to control. The principal dynamics of a biped is likened to an inverted pendulum, which once disturbed, quickly diverges from the upright position. The balance of a biped must be constantly maintained by stepping into the right place at the right time, and by using the torso as a reaction mass. Strong disturbances necessitate a timely response in order to act before the state of balance escalates beyond capturability. At the same time, walking is a rather complex whole-body motion. Excluding wrists, fingers, and the neck, typically a total of twenty degrees of freedom in the legs and the arms are relevant for the walking motion in a human-like kinematic chain. Consequently, a high-dimensional motion signal needs to be generated, and it has to be generated fast in order to be able to quickly absorb strong disturbances. The application to real hardware imposes additional challenges. The limited computational power of embedded systems, sensor noise, imprecise actuation, friction, backlash, and latency in the sensorimotor control loop can severely degrade the performance of a motion controller. It is no surprise that over the past decades, the replication of a well-balanced human-like walk has emerged as one of the most interesting challenges in robotics.

A bipedal gait is difficult to control due to its complexity and inherent instability.

The widespread state of the art covers basic walking on a flat surface without disturbances. Push recovery, walking on rough terrain, and agile footstep control are active research topics. The dominant strategy is to abstract from the complex body by representing it as a point mass with inverted pendulum dynamics. In most cases, the mathematically tractable Linear Inverted Pendulum Model is used to deduce controllers that steer and balance the pendulum in a way that the Zero Moment Point—the assumed pivot point of

The most successful methods use a low-dimensional physical model to control balance.

the inverted pendulum—stays within the boundaries of the support polygon. The trajectory of the point mass model is then transformed into a whole-body walking motion by mimicking the pendulum motion with the pelvis, connecting the pendulum base locations with smooth swing foot trajectories in Cartesian space, and computing the resulting motor commands using inverse kinematics. This approach works to some extent, but has not yet achieved the versatility and robustness of the human walk.

The method presented here fits a simple model to an open-loop walking motion and augments it with balance control.

The bipedal walk generation technique presented in this thesis differs conceptually from the state of the art. Instead of designing a low-dimensional model first and forcing a robot to follow its motion, we start with a central pattern-generated whole-body motion that can produce an open-loop stable walk. A low-dimensional inverted pendulum model is then fitted to match the open-loop motion. The fitted model augments the walking motion with balance control capabilities by controlling the timing and landing coordinates of footsteps in a non-intrusive way, leaving the execution of the stepping motions entirely up to the underlying pattern generator. The result is a robust omnidirectional gait controller that preserves the complexity of the walking motion. A restriction of the center of mass to a plane—a consequence of the limitations of the low-dimensional model—is not imposed on the robot. Walking with stretched knees is still possible.

Using a self-stable walking motion generator preserves the natural dynamics of the physical system.

Since we are able to make a robot walk without any additional feedback control, we assume that the central pattern-generated walk represents an instance of natural, self-stable dynamics of the biped. The augmentation with feedback control, which merely adjusts the step size and timing, does not derogate the natural dynamics. The fitting of the abstract model to real hardware motion data results in a forgiving controller. Precision requirements, that would otherwise need to be imposed when a robot is required to follow a model closely in order to preserve its theoretical stability, can be relaxed. Despite a high degree of imprecision and latency due to our unique setting of compliant, low-gain position controlled actuators, we achieved stable omnidirectional walking with arguably the strongest bipedal push recovery capabilities to date. It is the first bipedal gait controller that combines adaptive foot placement with a dynamic variation of the step timing. The motion generation with the Central Pattern Generator and the balance control using a low-dimensional model are both computationally efficient and suitable for high-frequency application on embedded systems.

The separation of motion and balance simplifies the learning of a balance controller.

Furthermore, the same technique of separating motion and balance lends itself to a machine learning approach. While the Central Pattern Generator hides the complexity of the whole-body walking motion, only a low-dimensional controller needs to be learned to control the timing and the coordinates of footsteps in order to maintain a balanced gait. Based on this approach, we implemented an online learning concept that learns during walking to adapt the sagittal step size in order to improve the balancing and reference tracking capabilities of the robot. During learning, a simple physical model suggests step size modifications with an estimated gradient, which speeds up the learning process to a level of performance where the controller is able to learn strong push recovery skills based on the experience of only a few failed steps. Moreover, the machine learning component can coexist with the analytic controller. The learning controller learns a corrective offset that is added to the sagittal step size output of the analytic controller and improves the overall walking performance. The Central Pattern Generator and the analytic controller constitute a fundamental initialization of the learning process in a way that the robot can already walk to some extent and has a concept of balance. This eliminates the two most critical aspects of online learning—starting with a constantly falling robot, which makes learning cumbersome and increases the risks of damage, and the necessity for the exploration of a large portion of the state space before a reliable performance is achieved.

A simple physical model is used to estimate a gradient that speeds up the learning process.

The remainder of this thesis is organized as follows. After reviewing related work in Chapter 2, we briefly introduce the bipedal robots in Chapter 3 that were used to carry out the experiments. In Chapter 4 we present an overview of the gait control framework in coarse detail. In Chapter 5, core principles of bipedal walking are introduced, which are distilled to the fundamental assumptions the analytic and the learned gait controllers are derived from. In the subsequent chapters the reader is guided through an in-depth presentation of the gait generation method. In Chapter 6, an open-loop Central Pattern Generator is introduced that is used to generate stepping motions. In Chapter 7, we elaborate on our state estimation method that simplifies the whole-body state to a low-dimensional point mass representation. In Chapter 8, an analytic footstep controller is presented that augments the open-loop motion generator with push recovery-capable feedback control. In Chapter 9, an online learning scheme for balanced gait control is introduced and our concept of learning a sagittal step size controller is covered. Experiments and discussions are presented inline. We close with a summary and conclusions.

2 RELATED WORK

The research topic of bipedal walking has been experiencing a continuous uprise of attention over the past decades. From an impenetrably large number of publications related to this topic, we selected the state of the art approaches with the largest impact in the robotics community. We divided the publications into two categories—analytic walk controllers, and learned controllers—and discuss these categories in the following sections.

2.1 ANALYTIC BIPEDAL WALKING

Zero Moment Point (ZMP) preview control [Kajita et al., 2003] is the most popular approach to bipedal walking. A number of pre-planned footsteps are used to define a future ZMP reference trajectory. A continuous Center of Mass (CoM) trajectory that minimizes the ZMP tracking error, the jerk of the CoM, and the deviation from terminal conditions at the end of the preview horizon, is then generated by solving a quadratic programming problem in a Model Predictive Control setting [Wieber, 2006]. The optimization is computationally expensive, but can be performed in real time. In theory, once a smooth and stable model is computed, a robot closely following the motion of the model should be stable too. By using the ZMP preview control scheme, high quality hardware [Kajita et al., 2010, Park et al., 2005] can walk reliably on flat ground as long as disturbances are small. Next generation gait controllers from the ZMP preview family [Diedam et al., 2008, Morisawa et al., 2010, Stephens and Atkeson, 2010] also consider foot placement in addition to ZMP control by including the footstep locations in the optimization process. Adaptive step timing has not yet been addressed by such methods. Perhaps the most capable version of the ZMP preview control family that includes adaptive foot-placement has been proposed by Urata et al. [2011]. Instead of optimizing the CoM trajectory for a single ZMP reference, a fast, iterative method is used to sample a whole set of lower quality ZMP/CoM trajectory pairs for three steps into the future. Triggered by a disturbance, the algorithm selects the best available motion according to given optimization criteria. Resampling during execution of the motion plan is not possible. The robot has to be able to track a fixed motion trajectory for the duration of the recovery. This algorithm was demonstrated to produce push recovery capabilities on a real robot. Highly specialized hardware was used to meet the precision requirements.

The most popular approach to bipedal gait control is the Zero Moment Point preview control algorithm.

Capture point based methods are simpler and can produce good results, but have not yet matured to adaptive foot placement.

The capture point is an appealing indicator of balance. The capture point [Pratt et al., 2006] is the location on the ground where a biped would theoretically need to step in order to come to a complete stop. Extensive work on stability analysis of bipedal systems has been presented by Pratt et al. [2012] based on the capture point dynamics. An analytical formalism was introduced to compute regions of N-step capturability for simple bipedal models that include a support area of non-zero size and a hip torque driven reaction mass. Englsberger et al. [2011] proposed the use of a capture point trajectory as a reference input for gait generation, instead of the ZMP. Since the capture point depends on the velocity of the pendulum mass, and not on the acceleration like the Zero Moment Point does, the system equations reduce to first order. The capture point approach is much simpler and faster to compute than ZMP preview control. A capture point based preview controller was demonstrated on Toro [Ott et al., 2010] to produce a walk of the same quality as the classic ZMP preview controller. Reactive foot placement and timing, however, have not been considered so far by this approach.

Tracking a low-dimensional model derogates natural dynamics and imposes high precision requirements.

A drawback of all of the aforementioned approaches is that the motion of a low-dimensional model is computed first, and then the robot is forced to follow its trajectory as closely as possible. This imposes precise position tracking requirements on the hardware in order to preserve the stability predicted by the model. Furthermore, a low-dimensional model strongly simplifies the walking motion by design. Following the model closely results in an unnatural, plane-restricted motion of the pelvis, typically with extensive use of bent knees.

Passive dynamic walkers can walk naturally without actuation and feedback control.

The inverse approach of starting with the motion itself originates from passive dynamic walking pioneered by McGeer [1990]. His experiments proved that not only control, but also actuation can be entirely removed from the system. The passive dynamics of legs with freewheeling joints is sufficiently stable to walk down a shallow slope. With a minimal amount of actuation to restore lost energy, passive walking on level ground is also possible [Anderson et al., 2005, Collins et al., 2001, Wisse and Frankenhuyzen, 2003]. The graceful motions of these bipedal constructions strongly resemble the human walk, and suggest that the core principle of biological gaits may also be passive dynamics with minimal control effort. Central pattern-generated walking adds actuation, but no control of balance. However, a small basin of attraction around the upright pose allows for open-loop stable walking when the floor is flat and external disturbances are not present. This includes some control of the walking direction and velocity. Position controlled motors with high gear ratios—a popular choice to actuate robots with Central Pattern Generator (CPG) driven gaits—unfortunately impede the passive dynamics of a system and trade energy efficiency for ease of control.

Central pattern generators add actuation, but no control of balance.

Interestingly, in the competitive environment of RoboCup, where humanoid robots play, win, and lose soccer games, open-loop walking is the dominant strategy. Due to the rapid change of hardware prototypes, software architectures, and scientific staff, the requirements on a walking algorithm shift towards adaptability to hardware modifications and simplicity of integration. Furthermore, the onboard computation capabilities of soccer robots set a limiting factor on the choice of walking algorithms, rendering the classic ZMP preview control algorithm less attractive.

Perhaps the most advanced RoboCup walk was presented for the Nao standard platform by Graf et al. [2009]. Based on the solution of a system of linear pendulum equations, the timing and trajectory of the pendulum motion is adjusted online in order to land the swing foot as closely as possible to a desired step size. This is one of very few examples that takes step timing into account. While a relatively weak open-loop core is present, the proposed feedback loop significantly increases the walking ability of the Nao robot.

The DARwIn-OP platform [Ha et al., 2013] has been very successful in the smallest humanoid robot class in the recent years. This capable hardware comes with a fast and reliable walk that has been described by Yi et al. [2011]. The core walking process has a strong similarity with ZMP preview control. Future footstep locations are placed in a queue and represent the future reference a few steps ahead. However, instead of the expensive CoM trajectory optimization that includes jerk minimization, the CoM trajectory is generated open-loop and in closed form using simple Linear Inverted Pendulum Model equations that do not limit the jerk. Swing foot trajectories are expressed as phase dependent trajectories in Cartesian space and are converted to joint motions using inverse kinematics.

Another remarkable CPG technique has been proposed by Dong et al. [2011]. Inspired by the capability of passive dynamic walkers to walk down shallow slopes, this approach shortens the swing leg before support exchange and creates a virtual downwards slope for the center of mass. The artificial shortening is reversed during the support phase and the dissipated energy is regained. Along with a simple, model-free and tunable algorithm for the generation of stepping motions, this approach comes with a mathematical framework to calculate optimal virtual slopes for desired walking velocities. This method has been successfully applied on robots of various sizes from the smallest to the largest of size classes of RoboCup.

The origin of the CPG used in this thesis was published by Behnke [2006]. Since then, the algorithm has evolved. The patterns have been simplified and extended with new capabilities that make the algorithm more flexible. It has been ported to new hardware, and compliant actuation has been experimented with. A detailed description of the latest version of the CPG [Missura and Behnke, 2013a] is included in Chapter 6.

The method presented in this thesis augments a Central Pattern Generator with balance control.

The focus of this thesis is the recently developed Capture Step Framework [Missura and Behnke, 2013b], which complements a CPG gait with balance control. It preserves the high-dimensional motion created by the CPG and allows for more natural walking with stretched knees. It has been demonstrated to generate a stable, omnidirectional walk with strong disturbance rejection capabilities on a real robot [Missura and Behnke, 2014b]. It computes both the balance control augmentation and the CPG efficiently in closed form, and is thereby suitable even for mobile devices with limited computational power.

By modeling virtual forces that keep the robot upright and pull it in the desired direction of locomotion, Pratt et al. [2001] created the Virtual Model Control approach. The virtual forces are mapped to the actuators of the robot such that the actuators produce the same trunk motion as the forces would. With this algorithm, the two-dimensional robot Spring Flamingo showed a fluid and natural looking walk that was robust enough to reject small disturbances and to walk up and down on slopes.

2.2 LEARNED BIPEDAL WALKING

When it comes to learning a bipedal gait controller, the leading state of the art is the simulation of artificial muscle control. Geyer and Herr [2010] introduced the concept of muscle reflex control and managed to produce a stable planar walk with a simulated biped that is actuated by Hill-type muscles and a set of simple reflexive feedback control laws. Wang et al. [2012] extended the concept to a 3D humanoid character and used Covariance Matrix Adaptation to optimize a large number of muscle activation and limb target position parameters. One of the main optimization criteria was the metabolic expenditure, which expresses the effort of transportation. The achieved result is a convincingly natural looking bipedal walk that locomotes at a fixed velocity and is robust to perturbations. Geijtenbeek et al. [2013] have extended the concept even further and included variable interaction points between the muscles and the skeleton to also be the subject of optimization. This way, a robust walk could be evolved for an arbitrary bipedal character. These methods are useful to generate natural looking sequences of computer animation. However, as the motions require hours to days on a large number of cores to optimize, and the precision of a high-frequency physical simulation, the applicability to real robots is not foreseeable in the near future.

When a real robot is in the loop, the feasible number of trials and the risk of damaging the hardware become limiting factors, and dictate that the learning process must reach a reliable walking performance after only a low number of experiences. Successful learning projects on real hardware to date typically start from either a fully functional motion generator, which is optimized for walking speed or stability during execution, or with a parameterized motion skeleton, and learn its parameters such that some form of stable walking is achieved within a feasible amount of iterations. Balancing concepts are investigated only in reduced settings, for example by learning how to stop in one step after a push.

Machine learning with real hardware is difficult due to the limited number of feasible repetitions.

Bipedal and quadrupedal gaits have been successfully optimized for walking speed using policy gradient methods [Hemker et al., 2009, Kohl and Stone, 2004]. With the same learning method, adjusted to a neuronal gait controller, the sagittal-only robot Runbot learned to walk with a high velocity and to cope with irregular terrain [Geng et al., 2006]. These experiments started from an open-loop stable, hand-designed gait.

Focusing on balance, Rebula et al. [2007] improved the reactive stepping of a simulated biped from a standing position by learning to step onto an offset from the capture point in order to bring the robot to a rest in an upright position. An orbital energy-based performance measure was introduced to evaluate the quality of an attempted step location, and a grid-based search was used to find a resolution-optimal solution in a local neighborhood.

A fast bipedal learning system was presented by Tedrake et al. [2005]. Using an online stochastic policy gradient estimation, the robot Toddler learned to walk on different surfaces in less than 20 minutes. The robot was designed in such way that it can passively walk down a slope without actuation. The success of this experiment can mostly be attributed to a strong simplification of the learning task to low-dimensional control of ankle actuation, imitating a passive dynamic gait without the need for a slope.

Yi et al. [2011] have investigated online learning on real hardware using a reinforcement learning method. This approach learns to optimize the input parameters of three biologically inspired disturbance rejection strategies that stabilize an open-loop gait trajectory generator. To make online learning in a real hardware setting feasible, the reinforcement learning was simplified by a discretization of the input space and the assumption that the control parameter space is restricted to parametric functions.

Morimoto and Atkeson [2009] used Gaussian processes [Morimoto et al., 2007] and Receptive Field Weighted Regression [Morimoto and Atkeson, 2007] to learn a Poincaré map that approximates the periodic dynamics of a biped. Using this map to select suitable actions, a policy gradient based reinforcement learning method was used to train bipedal gaits in simulation and on real robots. Upright walking with an unspecified walking velocity in the absence of disturbances was successfully achieved.

Relying on the concept of separating the generation of the walking motion from the control of balance, Missura and Behnke [2014a] reduced the difficulty of the learning task to learning only the low-dimensional control parameters of a CPG. By estimating an approximate gradient for the lateral step size with an inverted pendulum model [Missura et al., 2014] and for the sagittal step size with the pendulum-cart model Missura and Behnke [2014a], an efficient incremental learning approach learned strong push recovery capabilities on a simulated robot after only a few failed steps. The sagittal method has been implemented on a real robot and is presented as -part of this thesis.

3 THE ROBOTS

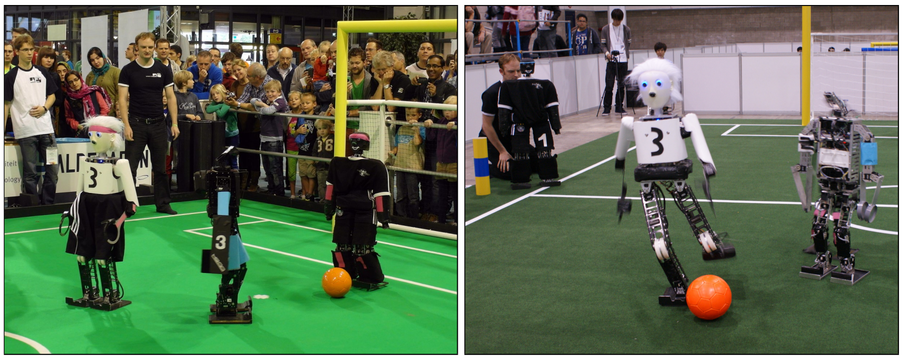


Figure 3.1: Scenes from RoboCup robot soccer games in the TeenSize class of the Humanoid League.

The bipedal gait generation framework presented in this thesis was designed with real hardware conditions in mind. Controlling real hardware is more difficult than controlling a simulated robot. While a physical simulation can offer access to high quality data, real robot control software must make do with the amount and quality of the data the sensor equipment can provide. Often, not all relevant physical quantities can be directly observed and must be inferred by combining different sources of information that are obscured by noise. The controller must succeed at its task using torque and velocity limited actuators prone to flawed response to control signals. These limitations apply to simulations only if they are explicitly included. Finally, particularly for tasks that involve balance constraints, such as the generation of a bipedal gait for a humanoid robot, a low-latency controller response is highly desired. Humanoid robots can be likened to an inverted pendulum that, once disturbed, quickly diverges away from the upright position. The quicker the controller response, the better the odds are for the robot to successfully avoid a fall after an unexpected disturbance. On a real robot, certain time delays needed for data transfers are unavoidable. The real world also cannot be paused to wait for a control iteration that has not finished computing.

Limited and noisy perception, imprecise actuation, and increased latency are challenges that make controlling a real robot more difficult than a simulated one.

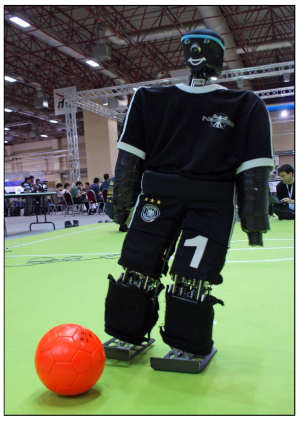
Two fairly similar, medium-sized humanoid robots with a height of a little over one meter were used to validate the proposed bipedal gait controller in real robot experiments. Certain properties of these robots provided a unique experimental environment that inevitably coined the design of the gait generation method presented in this thesis. For example, the robots are not equipped with foot pressure or ankle torque sensors. Therefore, it is not possible to determine the center of pressure underneath a foot, or whether a foot is in contact with the ground, in a straightforward manner. Other sources of information had to be used to infer ground contact, and the controller had to be designed in a way that it is robust to large errors in the ground contact estimation. Furthermore, the intelligent actuators of the robots support a compliant setting that permits shock absorption and a soft walking motion that tempers the stress on the motors. Unfortunately, this comes at the costs of imprecise control response and additional latency. However, since the compliant motion proved so beneficial for the hardware, and since it simulates conditions similar to serial elastic elements being present in the actuator drive train, it encouraged the development of components that specifically address lack of precision and high latency. Finally, the ability of the robots to tolerate the mechanical stress of a fall to a large extent without sustaining damage, and their low-cost, easy-to-repair mechanical construction, made it possible to experiment with the limitations of the presented gait controller. This is a rarely the case with robotic platforms, since contemporary humanoid robots of a relevant size, such as ASIMO [Hirai et al., 1998], HRP [Kajita et al., 2010], and HUBO [Park et al., 2005], are too fragile and too expensive to support overly dynamic experimentation, or even a single fall.

The robots used for experimentation are able to tolerate mechanical stress and are easy to repair.

The robots that were used to validate the bipedal gait controller carry the names Dynaped and Copedo. Their photographs and fact sheets are presented in Figure 3.2. Dynaped and Copedo can look back at a successful history in the annual RoboCup competitions. Having won five consecutive world championship titles from 2009 to 2013, and being awarded the Louis Vuitton Best Humanoid Award for outstanding performance in the years 2010 and 2012, these robots are among the most decorated robot soccer players worldwide. Figure 3.1 shows impressions of this robotic sport, where humanoid robots compete in fully autonomous games of soccer, played for two halves of ten minutes.

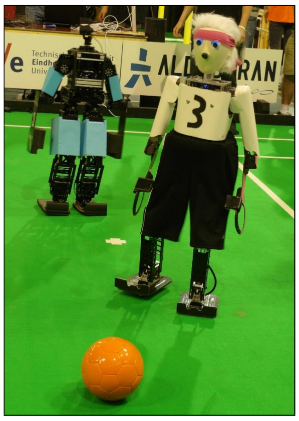
Both robots are constructed from milled aluminum and composite carbon fiber parts to keep the weight low. The robots carry a lithium polymer battery that provides the energy for an operation time of approximately 20 minutes. The robots are equipped with a Sony Vaio VGN-UX1XN ultra-mobile PC that takes charge of running the control software. The PC is fitted with a solid state disk, which reliably survives falls. A Freescale microcontroller board supports the data communication on the robot by distributing the actuator commands from the PC to all the actuators. It collects the motor position feedback from the actuators, trunk attitude data from an accelerometer and two gyroscopes, and transmits the sensor information as a bundled data packet via a USB connection to the PC. The somewhat involved communication effort loads the available bus to the limit of its

Dynaped



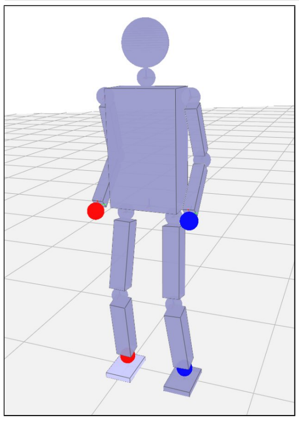
Total height:	105 cm
CoM height:	48.5 cm
Foot size:	24 cm x 15 cm
Weight:	7.5 kg
DoF:	13
CPU:	Intel 1.3 GHz Core Solo
Sensors:	dual accelerometer ADXL203 roll, pitch gyroscopes ADXRS
Motors:	Robotis Dynamixel EX-106, RX-64, RX-28

Copedo



Total height:	114 cm
CoM height:	53 cm
Foot size:	24.5 cm x 15.5 cm
Weight:	8 kg
DoF:	17
CPU:	Intel 1.3 GHz Core Solo
Sensors:	dual accelerometer ADXL203 roll, pitch gyroscopes ADXRS
Motors:	Robotis Dynamixel EX-106+, EX-106, RX-64, RX-28

Simon



Total height:	262 cm
CoM height:	122.8 cm
Foot size:	30 cm x 20 cm
Weight:	13.5 kg
DoF:	22
CPU:	-
Sensors:	-
Motors:	-

Figure 3.2: The humanoid soccer robots Dynaped and Copedo, and the simulated robot Simon were used to carry out the experiments in this thesis.

capacity and limits the loop rate of the control software to a frequency of 83.3 Hz.

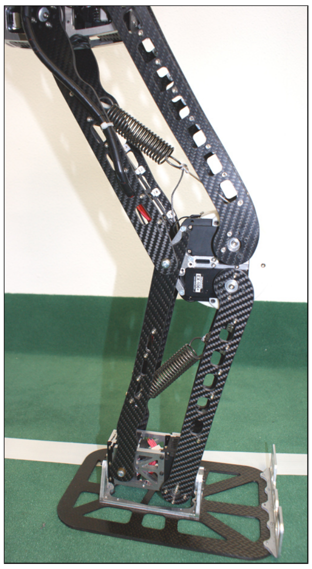


Figure 3.3: Parallel kinematic leg design.

The legs of both robots were constructed with a parallel kinematics structure, as shown in Figure 3.3. The parallel linkages in the thigh and in the shank mechanically force the knee joint to stay parallel to the trunk, and the foot plate to remain parallel to the knee joint. This construction lacks the degree of freedom in the ankle pitch, but provides additional passive stability. The robots are actuated by Robotis[1] Dynamixel intelligent actuators that simplify the task of motion generation with a position control interface. The position control is carried out by a high-frequency P-controller that modulates the pulse width of the motor voltage depending on the deviation of the current position of the output axle from the commanded position. When the P-controller is configured with a low gain, the actuator tolerates a larger error. This results in a relatively compliant actuator behavior with a soft feel, but it also adds a noticeable amount of latency. Over the years, the compliant actuation has proven to be an efficient protection for the motors and transmission gears. Incidents of broken gears drastically decreased. The tendency of the actuators to overheat was mitigated and the operation time of the robots increased. Figure 3.4 shows the commanded (tx) and measured (rx) motion trajectories of a knee actuator during in-place walking. Starting at approximately 1.5 seconds, a vertical downwards force was exerted by hand onto the hip of the robot. The actuator yields to the pressure and tolerates a certain amount of error. The

Compliant motion control is achieved by using a low-gain position control setting provided by the actuators.

1 http://www.robotis.com

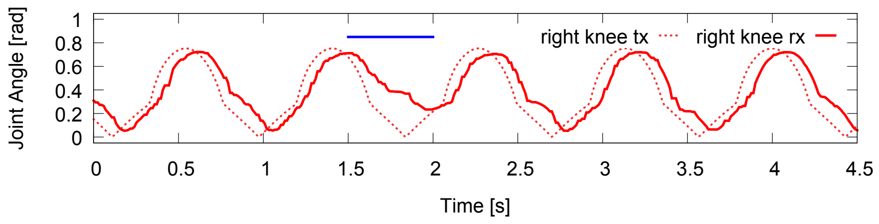


Figure 3.4: Commanded (tx) and measured (rx) joint angles of the right knee during walking in place. To demonstrate the joint elasticity, the robot was pushed down on the right hip at 1.5 seconds. The blue bar indicates the time and duration of the push. The position tracking error increases significantly at this point.

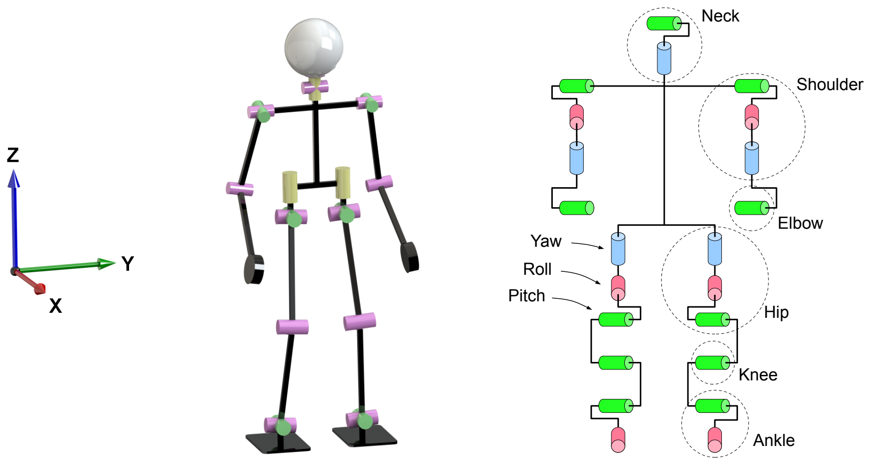


Figure 3.5: A generic kinematic chain that applies to most humanoid robots.

compliant actuator setting combines best with the parallel kinematic construction. In experiments with robots that had an articulated ankle pitch axis, a stiffer configuration had to be used for the pitch motors. The compliance of the roll motors could still be set to a high value.

The third robot introduced in Figure 3.2 is a simulated one. It was given the name Simon. This robot was frequently used to perform theoretical experiments in a Bullet 2.82 physics simulation. The same software was used to control the walk of the simulated robot and the real robots, with the only difference that in simulation we were able to use a higher control loop frequency of 100 Hz. Notably, Simon is significantly larger than the two real robots Dynaped and Copedo. For numerical reasons it is easier for the Bullet Physics engine to process the kinematic chain when the distances between the constraints that hold the rigid body segments together are large. The physical simulator produced the same qualitative physical behavior that we observed with the real robots. The difference in size did not seem to have a notable influence.

The mechanical construction of a robot implicitly defines a fixed order of rotations in the kinematic chain. The kinematic order of the axes of rotation matters, because a rotation about one axis also rotates all subsequent axes in the chain. The kinematic chain of Simon is shown in Figure 3.5. This is the kinematic order we assume for all subsequent computations. For all joints, we assume a right-handed coordinate frame with the convention that the x-axis points in the forward direction of the robot and corresponds to the axis of the roll rotation. The z-axis—the axis of the yaw rotation—points upwards. The y-axis completes the right-handed coordinate frame and is the axis of the pitch rotation. As far as the order of the degrees of freedom is concerned, the kinematic layout shown in Figure 3.5 is very common amongst humanoid robots. For example, the hip joint is typically designed in a yaw-roll-pitch order. The actuators of Dynaped

The kinematic chain shown in the figure above, and an x-forward, right-handed coordinate system are our underlying assumptions for kinematic computations.

and Copedo are also arranged according to this kinematic chain, except that some of the degrees of freedom are missing. Due to the parallel kinematics, Dynaped and Copedo both lack the ankle pitch axis. The neck joints of Dynaped and Copedo also only have the yaw degree of freedom, but not the pitch. The shoulders of Copedo do not have the yaw degree of freedom and the shoulders of Dynaped only have a pitch degree of freedom. Dynaped also does not have elbows. The missing degrees of freedom of the real robots are simply ignored during motion execution. This is possible because the motion of the neck joint is not relevant for walking, the motion of the arms uses the shoulder pitch degree of freedom, which both robots have, and the parallel leg kinematics of the robots mechanically moves the ankle pitch the same way the gait motion generator would do. For the generation of stepping motions, the definition of the order of the rotational axes is sufficient. The sizes of the rigid body elements only become relevant for the pose reconstruction presented in Section 7.2.

In the following chapter, an overview of the gait generation framework is presented, where all relevant components are introduced in reduced detail. This complete picture is used as a guideline when descending into the detailed descriptions of each component in later chapters.

4 OVERVIEW

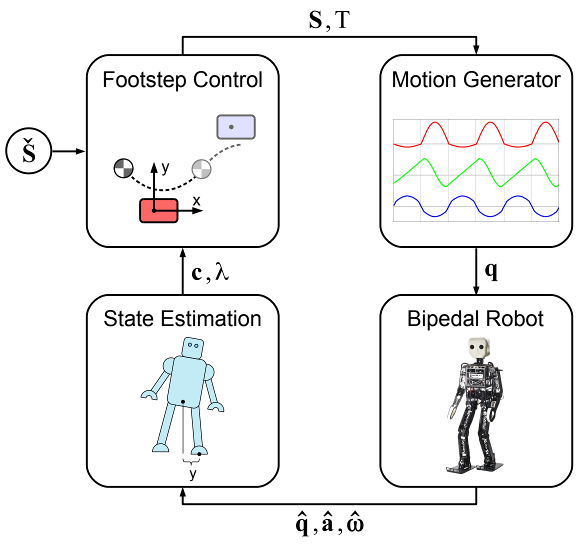


Figure 4.1: Overview of the Capture Step Framework. The State Estimation component (bottom left) reconstructs the pose of the robot from the sensors reporting the joint angles $\hat{\mathbf{q}}$, the inertial acceleration $\hat{\mathbf{a}}$ of the trunk, and the angular velocity $\hat{\boldsymbol{\omega}}$ of the trunk. The Footstep Control module (top left) uses the point mass representation $\mathbf{c}$ and the support foot indicator λ to compute the location $\mathbf{S}$ and the timing T of the next footstep in order to track the desired step size $\check{\mathbf{S}}$ while maintaining balance. The Motion Generator (top right) executes a timed whole-body stepping motion towards the commanded footstep coordinates and generates the joint position targets $\mathbf{q}$.

The bipedal gait generation method presented in this thesis is called the *Capture Step Framework*. Figure 4.1 illustrates the structure of the framework. It is organized in the typical circular layout of a control loop with functionally separable modules. The layout is the result of a unique approach to controlling an open-loop Central Pattern Generator (CPG) using footstep coordinates and timing. The robot itself (bottom right) is part of the loop. It receives motor targets from the control software and provides sensor data about its internal state. Three logical software components can be identified: State Estimation, Footstep Control, and Motion Generation.

Our gait control method is organized in a framework with functional components for state estimation, balance control, and motion generation.

A higher control instance can command a reference step size $\check{\mathbf{S}}$ that the biped should produce. This choice is motivated by the fact that footstep planning [Chestnutt et al., 2005, Hornung et al., 2012] is a gradually improving, versatile method to command a robot where to walk. A footstep plan can be used to simply encode constant velocity walking on a flat surface, but has the flexibility to scale up to careful

The control input is a desired step size for compatibility with footstep planning.

stepping onto constrained locations in cluttered environments, stepping over obstacles, and using elevated footholds in rough terrain.

Using the joint angles $\hat{q}$, the inertial acceleration $\hat{\alpha}$ of the trunk, and the angular velocity $\hat{\omega}$ of the trunk, as measured by the sensors of the robot, the State Estimation module reconstructs the whole-body robot pose. The pose reflects the kinematic configuration of the robot, i. e. the angle of its body parts relative to their parent, and the attitude of the trunk in the world coordinate frame. From the reconstructed pose, the motion of a fixed point on the body frame is tracked and used as a low-dimensional representation of balance. Coordinates and velocities of this fixed point—referred to as the Center of Mass (CoM) state $c = (c_x, \dot{c}_x, c_y, \dot{c}_y)$—are determined in the sagittal (x, forwards) and the lateral (y, sidewards) directions with respect to the support foot $\lambda \in \{-1, 1\}$. We assign the sign -1 to the left foot and the sign 1 to the right foot. The CoM state vector $\mathbf{c}$, the sign λ of the support foot, and the desired step size $\check{\mathbf{S}}$, are the inputs into the Footstep Control module, where a Linear Inverted Pendulum Model (LIPM) is used to reason about the location $\mathbf{S}$ and timing T of the next footstep in order to keep the center of mass balanced while obeying the desired step size $\check{\mathbf{S}}$ as closely as possible. We hereby term this task *footstep control*. The step size $\mathbf{S}$ and the step time T are passed on to the Motion Generator module as step parameters. The Motion Generator concretizes the step into a whole-body stepping motion. The whole-body motion trajectory is conveniently expressed as a signal of joint angles $\mathbf{q}$, which yields the next target position for each of the position controlled servo motors of the robot. The control loop executes at a frequency of approximately 100 Hz[1].

We term the task of tracking a desired step size while maintaining balance "footstep control".

At this point, it is advisable to watch the demonstration video [1] in the List of Videos, where bipedal robot Dynaped demonstrates the capabilities of the Capture Step Framework. It enables agile, omnidirectional walking with recovery capabilities from a number of different types of disturbances, such as pushes, unexpected objects under the foot, and collisions. The video material presented in the List of Videos is referenced throughout the thesis to demonstrate experiments and specific features.

In the following chapter, we introduce underlying principles we observed from walking bipedal robots. These are coarse physical principles that help simplify the task of modeling a balance controller. We subsequently turn our attention to the whole-body motion generation component.

1 The bipedal robots Dynaped and Copedo that were used to carry out the real hardware experiments are run with a control loop frequency of 83.3 Hz.

5 PRINCIPLES OF BIPEDAL WALKING

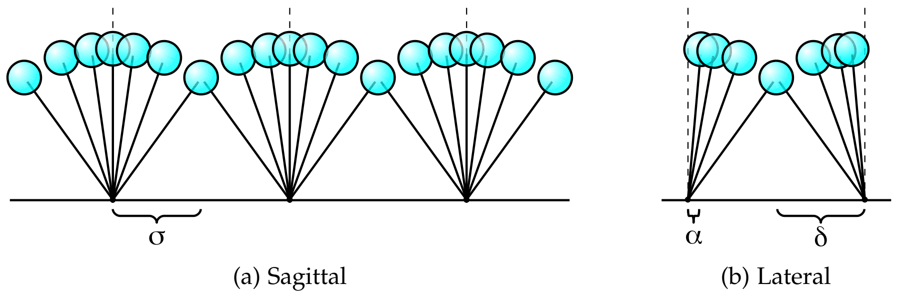


Figure 5.1: Stick diagrams of a compass gait. (a) In the sagittal direction, the center of mass crosses the pendulum pivot point in every gait cycle. (b) In the lateral direction, the center of mass oscillates between the pivot points. The parameter annotations highlight characteristic quantities. σ is the maximum center of mass displacement in sagittal direction. α marks the minimum distance between the center of mass and the pivot point at the lateral step apex. δ shows the lateral support exchange location of the center of mass in the center of a comfortable stride width.

The inverted pendulum-like dynamics of the human walk has been long known to be an economical principle of locomotion [Kuo et al., 2005]. Vaulting over a stretched leg and exploiting the ballistic motion of the swing leg results in an almost effortless way of transportation, suitable for covering large distances. However, the energy efficiency comes at the cost of stability. Bipeds constantly have to use small corrections to remain within a comfortable cycle, and well-placed footsteps to react to large disturbances.

The walking motion can be decomposed into a sagittal and a lateral motion.

For the purpose of designing a balance controller for bipedal walking it is beneficial to understand the nature of the walking motion itself. Figure 5.1 shows stick diagrams of the idealized pendulum motion projected onto the sagittal plane and the frontal plane. This orthogonal decomposition is commonly used to analyze the walking motion, since other than a shared time of support exchange, the sagittal and the lateral motion do not seem to have a significant influence on each other. Interestingly, the sagittal and lateral motions exhibit strongly distinct behaviors. In the sagittal plane, the center of mass crosses the pivot point of the pendulum in every gait cycle, while in the frontal plane, the center of mass oscillates between the support feet and never crosses the pivot point.

In the lateral direction the set of reachable footholds is constrained, as placing the swing leg on the other side of the support leg would

The lateral direction is more constrained than the sagittal direction.

require the legs to cross, which humanoid robots are typically not able to do. Consequently, tipping over sideways in the direction of the support leg is a critical situation that requires an exceptionally challenging sequence of motions to recover from. The small lateral distance at the apex of the step between the pivot point and the center of mass (annotated with the parameter α in Figure 5.1b) provides a narrow margin for error. As long as the center of mass returns, a step can be taken. Otherwise, the biped tips over and cannot reasonably step. In the sagittal direction, however, viable footholds are only constrained by the kinematic range of the legs. The walker can comfortably place the swing leg in front of, or behind, the support leg.

Sidestepping is a cumbersome motion of alternating step sizes.

Walking in the sagittal direction is a determined and fluid motion. The center of mass travels straight forward (or backward) on an arched trajectory over the stretched respective support leg. Walking in the lateral direction, however, is cumbersome. In an alternating manner, the walker takes a large step with the leading leg, followed by a small trailing step, without crossing the legs. The stride width of the trailing step is approximately the same as the comfortable stride width that emerges when not walking in the lateral direction. This characteristic stride width is denoted δ, and is indicated in Figure 5.1b as the center point of the comfortable stride. Humans, although able to cross their legs, usually abstain from sidestepping and prefer to follow the geometric paths of non-holonomic systems, such as cars steering their way to their target along Clothoids [Laumond et al., 2011].

The lateral oscillation of the center of mass is sensitive to disturbances.

The perpetual lateral oscillation of the center of mass, a consequence of the absence of static stability, appears to be the primary determinant of the step timing. Disobeying the right timing can quickly destabilize the lateral swing. The video experiment [3] demonstrates how a small disturbance of the lateral oscillation can be sufficient for the system to destabilize and fall. In the sagittal direction, timing appears to be less crucial. Following from the law $v \approx 2\sigma\Omega$, the velocity v of the center of mass can be the result of an infinite amount of step frequency Ω and stride length 2σ combinations. Therefore, the biped can flexibly accommodate small variations in timing with a change of the stride length, e. g. take a short and quick step or a long and slow one, and not disturb the CoM velocity. The observed flexibility in the sagittal direction in contrast to the constrained motion in the lateral direction coincides with the fact that humans invest substantially more effort into lateral control [Bauby and Kuo, 2000, Kuo, 1999].

Stretched legs imply symmetrical steps.

From a biomechanical perspective, walking with stretched support legs and with the center of mass traveling on circular arcs (rather than on a horizontal line) is a causal effect of the energetic cost of locomotion [Kuo, 2007]. From a modeling perspective, stretched knee walking leads to a convenient consequence. If in the moment of the heel strike both legs are stretched out, they are of nearly equal lengths.

Hence, the center of mass is approximately above the center point between the two pendulum pivot points. We refer to this configuration as symmetrical stepping.

Based on these observations, we make the following assumptions that all our mathematical and algorithmic models of a bipedal walk are based on:

- Synchronized at the time of the support exchange, the sagittal and the lateral motions can be treated as uncoupled entities.
- Control priority of the time of the support exchange is allocated to the lateral direction. The primary determinant of the time of the support exchange is the moment when the center of mass reaches a nominal support exchange location in the center of the stride width.
- Variations in timing are accommodated by the choice of sagittal stride length.
- A constant lateral apex distance (α) should be maintained in order to preserve lateral stability and to avoid tipping over.
- The final configuration of a step in the moment of the heel strike is symmetrical. The center of mass is in the middle between two pivot points.

Naturally, these assumptions do not capture the full complexity of the human gait. However, they simplify the mathematical modeling of a balance controller, yet preserve the possibility to walk with stretched knees.

In the next chapter, we introduce a central pattern generator that can produce an open-loop stable bipedal gait with stretched legs and symmetrical steps. Then, we turn our attention to an analytic step controller that introduces feedback techniques into the gait generation process, based on the assumptions listed above.

6 STEP MOTION GENERATOR

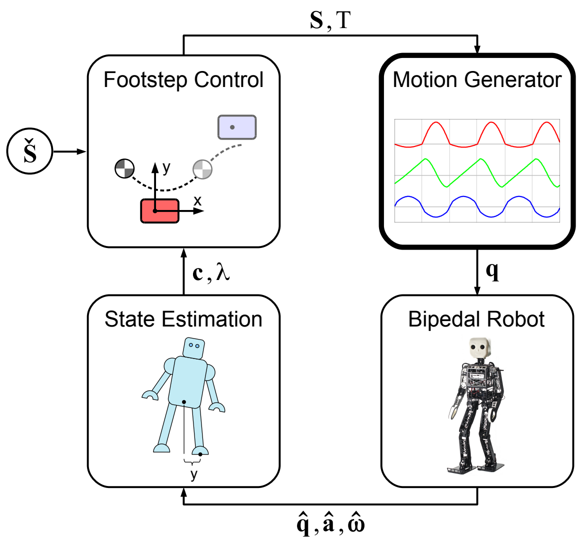


Figure 6.1: The Motion Generator component is a central pattern generator that generates the motor targets $\mathbf{q}$ of a whole-body stepping motion. The step size $\mathbf{S}$ and the landing time T are provided as control input.

Often a small basin of attraction allows for self-stable walking.

The challenge of maintaining balance on rough terrain and in the presence of strong disturbances remains a difficult task for bipedal robots. In undisturbed flat-floor environments, however, a small basin of attraction around the upright pose is sufficient to support open-loop walking. Numerous teams in the Humanoid League of RoboCup rely on this technique, and are able to implement a relatively stable and dynamic walk for bipedal robots, providing for exciting games of robot soccer. Team NimbRo[1] is one of the most successful teams in the Humanoid League to date. Playing with self-constructed prototypes, the team managed to win the KidSize competitions in 2007 and 2008 and the TeenSize competitions from 2009 to 2013. A central pattern-generated gait that has been successfully adapted to a number of prototypes of varying sizes is one of the strengths of team NimbRo. Table 6.1 shows robots that this gait has been adapted to over the years. The motion patterns used by team NimbRo were originally proposed by Behnke [2006]. Since then, the algorithm has evolved. The patterns have been extended with new capabilities along with new configuration parameters that make the algorithm more flexible. It has been adopted to new hardware and compliant actuation has been experimented with. The latest version of the CPG has been pub-

1 http://www.nimbro.de

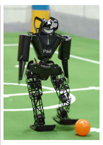
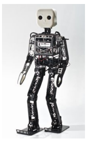
								
Class	KidSize			TeenSize			NimbRo-OP	
Year	2006	2007	2008	2007	2008	2011	2013	2014
Name	Paul	Lothar	Steffi	Robotina	Dynaped	Copedo	Po	P1
Size	60 cm	60 cm	60 cm	122 cm	105 cm	114 cm	95 cm	90 cm
Weight	2.9 kg	4 kg	3.5 kg	9 kg	7.5 kg	8 kg	6.6 kg	6.6 kg
DOF	20	20	17	21	13	17	20	20
Cameras	2	3	3	3	3	1	1	1
Sensors	2-axis acc, gyro			2-axis acc, gyro			3-axis acc, gyro	
CPU	XScale 520 MHz	Intel Core Solo ULV 1.33 GHz					AMD-450 2×1.6 GHz	

Table 6.1: Humanoid prototypes of the robot soccer team NimbRo. Source: http://www.nimbro.de

lished by Missura and Behnke [2013a]. This CPG has been selected as the Motion Generator component for the Capture Step Framework.

The NimbRo CPG generates an omnidirectional walk with stretched knees.

The NimbRo CPG generates an omnidirectional walk that allows a humanoid robot to step in the sagittal, lateral, and rotational directions. The possibility to independently and simultaneously control the step size in these three directions gives rise to a relatively agile and dynamic walk that has been a key advantage in robot soccer games since the year of its first application in 2006. Today, an omnidirectional walk is standard amongst humanoid RoboCup teams. Interestingly, the NimbRo gait is able to recover from mild disturbances, such as light pushes and stepping on small objects on the floor, even though it is completely open-loop. The CPG offers a fair number of parameters to tune the motion patterns, enabling flexible adaptation to individual robot prototypes. The walking motion keeps the knees almost completely stretched during the support phase. The constant leg length during the support phase results in a non-level CoM trajectory. This is a desirable feature that reduces the energetic cost of transportation during walking. Furthermore, the CPG can be combined with optional low-gain position control actuation for softer leg motions.

In the following sections we introduce the layered architecture of the motion generator and descend into a detailed explanation of each layer. Then, we show experiments to analyze the open-loop stability that was achieved with a simulated and a real robot using the NimbRo CPG.

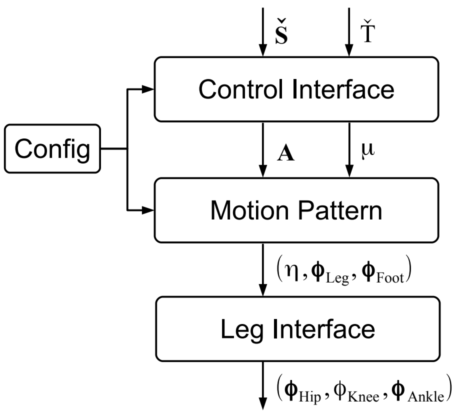


Figure 6.2: Layers of the NimbRo gait generator. The Control Interface receives a step size target $\check{\mathbf{S}}$ and a desired step time $\check{T}$ from a higher layer and translates them into a leg swing activation vector $\mathbf{A}$, and a motion phase μ. The Motion Pattern layer generates phase dependent periodic motion signals in an abstract leg position parameter space, that is then mapped to joint targets by the Leg Interface.

6.1 THE LAYERS OF THE MOTION GENERATOR

The gait generator can be decomposed into three hierarchical layers of a control interface, pattern generator, and abstract kinematic interface.

The NimbRo gait generator algorithm can be represented by three logical layers, as illustrated in Figure 6.2. The topmost layer is the Control Interface. It receives a foot-to-foot step size vector $\check{\mathbf{S}} = (\check{S}_x, \check{S}_y, \check{S}_\psi)$ with sagittal, lateral, and rotational coordinates of the swing foot relative to the support foot, and a desired step time $\check{T}$ from a higher control instance. The Control Interface translates the desired step size $\check{\mathbf{S}}$ into a swing amplitude activation vector $\mathbf{A} = (A_x, A_y, A_\psi) \in [-1, 1]^3$ with roll, pitch, and yaw parameters. The swing activation vector $\mathbf{A}$ determines the leg swing amplitude—and thus the step size—during walking. The Control Interface also maintains a continuous flow of the motion phase $\mu \in [-\pi, \pi)$, which is computed based on the target step time $\check{T}$. The Motion Pattern layer generates periodic motion signals that produce omnidirectional stepping motions, targeted to match the input step size at the commanded time. The motion pattern generation is aided by an abstract kinematic interface constituted by intuitive leg pose parameters, such as the angle of the leg relative to the trunk, and the extension of the leg. These parameters are translated into joint targets by the Leg Interface abstraction layer. A set of configuration variables is an integral part of the algorithm that allows easy hardware adaptation. The three layers are best explained in more detail in a bottom-up order.

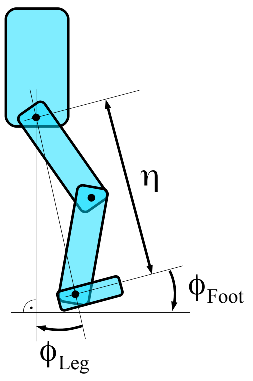


Figure 6.3: The Leg Interface encapsulates leg pose control with three abstract parameters: the leg extension η, the leg angle $\boldsymbol{\phi}_{\mathtt{Leg}}$, and the foot angle $\boldsymbol{\phi}_{\mathtt{Foot}}$.

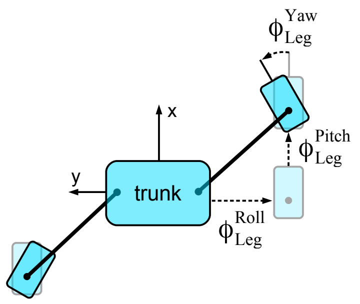
6.2 ABSTRACT KINEMATIC INTERFACE

The CPG uses an abstract kinematic interface to control the joints of the robot.

The presented gait generation algorithm is entirely based on a kinematic abstraction layer that we dubbed the *Leg Interface*. The Leg Interface exhibits three abstract parameters to control the pose of a leg. The meaning of the parameters is illustrated in Figure 6.3. A leg extension parameter η determines the distance between the foot and the trunk and allows the leg to be extended and retracted like a prismatic joint. A leg angle parameter $\boldsymbol{\phi}_{\mathtt{Leg}} = (\phi^{\mathtt{Roll}}_{\mathtt{Leg}}, \phi^{\mathtt{Pitch}}_{\mathtt{Leg}}, \phi^{\mathtt{Yaw}}_{\mathtt{Leg}})$ determines the rotation of the leg with respect to the trunk. The leg can be rotated in the roll, pitch, and yaw directions. To roll or pitch the foot, the foot angle parameter $\boldsymbol{\phi}_{\mathtt{Foot}} = (\phi^{\mathtt{Roll}}_{\mathtt{Foot}}, \phi^{\mathtt{Pitch}}_{\mathtt{Foot}})$ is used to determine the rotation of the foot with respect to the trunk. The coordination of the hip, knee, and ankle joints that supports the use of this interface is computed automatically by the Leg Interface.

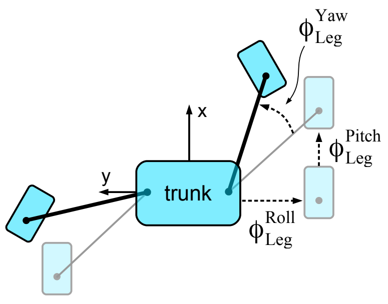
The Leg Interface parameters span a convenient orthogonal space for the design of motion patterns.

The parameter space of the Leg Interface offers a rather intuitive way to move a leg. The parameters encode motion components that a robot would naturally perform during walking and simplify the task of pattern generation. For example, lifting the leg up at the beginning of the swing phase, and stretching it out shortly before the heel strike, can be achieved using the leg extension parameter η. Swinging the leg back with a stretched knee during the support phase is achieved by keeping the leg extension parameter η constant, and using the leg angle parameter $\phi^{\mathtt{Pitch}}_{\mathtt{Leg}}$ to smoothly modify the leg pitch angle. It is much easier to express these typical leg swing motions with Leg Interface parameters than in end-effector coordinates of an inverse kinematics-based motion interface, where the arcs that the feet describe relative to the trunk during a stretched knee walk would have to be explicitly generated first. Conveniently, the input parameters η, $\boldsymbol{\phi}_{\mathtt{Leg}}$, and $\boldsymbol{\phi}_{\mathtt{Foot}}$ operate independently of each other. For example, if we shorten the leg using the leg extension parameter η, the Leg Interface computes the joint angles in a way that the leg angle and the



$\phi^{\text{Yaw}}_{\text{Leg}}$
x
y
trunk
$\phi^{\text{Pitch}}_{\text{Leg}}$
$\phi^{\text{Roll}}_{\text{Leg}}$

(a) Roll-pitch-yaw



$\phi^{\text{Yaw}}_{\text{Leg}}$
x
y
trunk
$\phi^{\text{Pitch}}_{\text{Leg}}$
$\phi^{\text{Roll}}_{\text{Leg}}$

(b) Yaw-roll-pitch

Figure 6.4: (a) The leg angle parameter $\boldsymbol{\phi}_{\text{Leg}}$ is interpreted in a roll-pitch-yaw order. This way, the pitch and roll components of the leg angle correspond to the x and y-coordinates of the foot in the trunk frame, and the trunk is in the center between the feet when both legs are positioned with the same parameter values, but with opposing signs. (b) The yaw-roll-pitch order of the hip rotation implied by the mechanical construction does not provide these advantages.

foot angle remain untouched. Similarly, modifying the angle of the leg with the leg angle parameter $\boldsymbol{\phi}_{\text{Leg}}$ does not have an influence on the extension of the leg and also preserves the foot angle. If we were to move the knee joint directly, the leg extension, the leg angle, and the foot angle would all be modified at once in a non-trivial manner.

Formally, the Leg Interface is a function

$$\left(\boldsymbol{\phi}_{\text{Hip}}, \phi_{\text{Knee}}, \boldsymbol{\phi}_{\text{Ankle}}\right) = \mathcal{L}\left(\eta, \boldsymbol{\phi}_{\text{Leg}}, \boldsymbol{\phi}_{\text{Foot}}\right) \tag{6.1}$$

that encapsulates the mapping of the abstract parameters to joint angles $\boldsymbol{\phi}_{\text{Hip}} = (\phi^{\text{Roll}}_{\text{Hip}}, \phi^{\text{Pitch}}_{\text{Hip}}, \phi^{\text{Yaw}}_{\text{Hip}})$ for the hip joint which has three rotational axes in the roll, pitch, and yaw directions, ϕ_{Knee} for the knee joint which has only a pitch axis, and $\boldsymbol{\phi}_{\text{Ankle}} = (\phi^{\text{Roll}}_{\text{Ankle}}, \phi^{\text{Pitch}}_{\text{Ankle}})$ for the ankle joint which has pitch and roll axes. The joint angles are computed using the equations

$$\begin{bmatrix} \phi'^{\text{Pitch}}_{\text{Leg}} \\ \phi'^{\text{Roll}}_{\text{Leg}} \end{bmatrix} = R(-\phi^{\text{Yaw}}_{\text{Leg}}) \begin{bmatrix} \phi^{\text{Pitch}}_{\text{Leg}} \\ \phi^{\text{Roll}}_{\text{Leg}} \end{bmatrix}, \tag{6.2}$$

$$\zeta = \arccos(1-\eta), \tag{6.3}$$

$$\phi^{\text{Yaw}}_{\text{Hip}} = \phi^{\text{Yaw}}_{\text{Leg}}, \tag{6.4}$$

$$\phi^{\text{Roll}}_{\text{Hip}} = \phi'^{\text{Roll}}_{\text{Leg}}, \tag{6.5}$$

$$\phi^{\text{Pitch}}_{\text{Hip}} = \phi'^{\text{Pitch}}_{\text{Leg}} - \zeta, \tag{6.6}$$

$$\phi_{\text{Knee}} = 2\zeta, \tag{6.7}$$

$$\phi^{\text{Pitch}}_{\text{Ankle}} = \phi^{\text{Pitch}}_{\text{Foot}} - \phi'^{\text{Pitch}}_{\text{Leg}} - \zeta, \tag{6.8}$$

$$\phi^{\text{Roll}}_{\text{Ankle}} = \phi^{\text{Roll}}_{\text{Foot}} - \phi'^{\text{Roll}}_{\text{Leg}}. \tag{6.9}$$

The order of the motors in the kinematic chain of the robot (shown in Figure 3.5) implies a yaw-roll-pitch order of the rotation of the hip

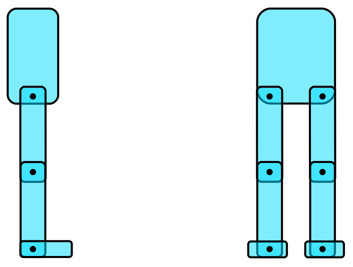


Figure 6.5: The Zero Pose is a mechanically defined pose of the robot where the legs are fully extended, parallel to the trunk, and parallel to each other. In the Zero Pose, The feet are orthogonal to the legs.

The roll-pitch-yaw rotation order of the leg angle centers the trunk in between the feet.

joint. However, we rotate the pitch and roll components of $\boldsymbol{\phi}_{\mathtt{Leg}}$ with a counter-clockwise rotation matrix R in equation 6.2 by the negative leg yaw and define the interpretation of the leg angle parameter $\boldsymbol{\phi}_{\mathtt{Leg}}$ to be in a roll-pitch-yaw order. This interpretation bears two advantages that are illustrated in Figure 6.4. Typically, the legs move in a symmetrical way during walking, such that $\boldsymbol{\phi}^{\mathtt{Left}}_{\mathtt{Leg}} \approx -\boldsymbol{\phi}^{\mathtt{Right}}_{\mathtt{Leg}}$, i. e. the left leg is positioned using leg angle parameters that have the opposite signs of the leg angle parameters for the right leg. When using the roll-pitch-yaw interpretation as shown in Figure 6.4a, the $\phi^{\mathtt{Yaw}}_{\mathtt{Leg}}$ parameter rotates the foot around the ankle joint, rather than the entire leg around the hip joint, and the trunk is always approximately in the center between the feet. If we were to use a yaw-roll-pitch interpretation of the leg angle parameter as shown in Figure 6.4b, the yaw rotation of the legs around the hip joints with opposing signs would not place the trunk in the center between the feet. The second advantage is that since only small angles are used to activate the roll and pitch leg angle parameters, the $\phi^{\mathtt{Pitch}}_{\mathtt{Leg}}$ parameter corresponds to the x-coordinate and the $\phi^{\mathtt{Roll}}_{\mathtt{Leg}}$ parameter corresponds to the y-coordinate of the foot in the reference frame of the trunk. This is highly convenient for the conversion of the input step size $\check{\mathbf{S}}$ to the swing amplitude vector $\mathbf{A}$ in the Control Interface. This is discussed in Section 6.4.

The leg extension parameter η is expected to lie in the unit interval $[0, 1]$. The leg and foot angle parameters can have arbitrary values as far as the Leg Interface is concerned. Joint angle limitations are enforced on a deeper level on a per joint basis.

We assume that the thigh and the shank of the robot are of equal lengths. Humanoid robots are frequently constructed this way. In the case of a different kinematic configuration, the Leg Interface equations (6.3-6.9) need to be adjusted accordingly. Please note that despite this assumption, the motion abstraction layer is essentially model free. Unlike for inverse kinematics, the actual sizes of the body segments do not need to be known.

We follow the convention that if all Leg Interface parameters are set to zero, the Leg Interface computes a value of zero for all joint angles. We define that in this *Zero Pose* where all joint angles and all Leg Interface parameters are zero, the legs are fully extended, parallel to the trunk and to each other, and the feet are orthogonal to the legs. The Zero Pose is illustrated in Figure 6.5. This is a safe and reproducible pose, well-suited for calibration. The definition of the Zero Pose implies that a leg extension of $\eta = 0$ is interpreted as a fully extended leg and a leg extension of $\eta = 1$ is interpreted as a fully retracted leg.

6.3 MOTION PATTERN

The NimbRo gait pattern is a combination of rhythmic activation signals that encode periodic leg-lifting and leg-swinging motions. The leg swing amplitude is determined by the swing activation vector $\mathbf{A} = \left(A_x, A_y, A_\psi\right) \in [-1, 1]^3$ with parameters for the roll, pitch, and yaw directions. The motion signal oscillation is driven by a motion phase $\mu \in [-\pi, \pi)$. The motion phase is incremented smoothly with a small increment in every iteration of the main control loop. When the next increment would set the motion phase to a value $\mu \geqslant \pi$, the motion phase is reset to $\mu = -\pi$. The output of the pattern generator is a set of Leg Interface parameters. The Leg Interface conveniently allows us to design leg angle and leg extension trajectories instead of having to think about coordinated motions on the single joint level.

The pattern generator is a combination of periodic motion primitives. The same pattern is used to generate the motion for both legs.

The walking motion pattern can be subdivided into motion primitives. The most important motion primitives for leg lifting, leg swinging, and hip swinging motions are shown in Figure 6.6. The same motion pattern is used to generate the motion for both legs. The pattern generator is executed twice, once for the right leg with the leg phase $\nu = \mu$ and once for the left leg with a time-shifted leg phase $\nu = \mu + \pi$. In some cases the sign $\lambda \in \{-1, 1\}$ of the leg (left or right) is used to determine the direction of a rotation, depending on which leg the pattern is generated for.

In the following sections we introduce the motion primitives in detail, and refer to Figure 6.6 where appropriate. The motion primitives are summated to the final step motion pattern that produces the Leg Interface parameters. After the presentation of the pattern generation process, we provide an annotated set of configuration parameters in Table 6.2, as used for bipedal robot Dynaped, the robot that has been used to carry out most of the experiments in this thesis. We denote motion primitives with the letter $\mathbf{P}$ and configuration variables with the letter K.

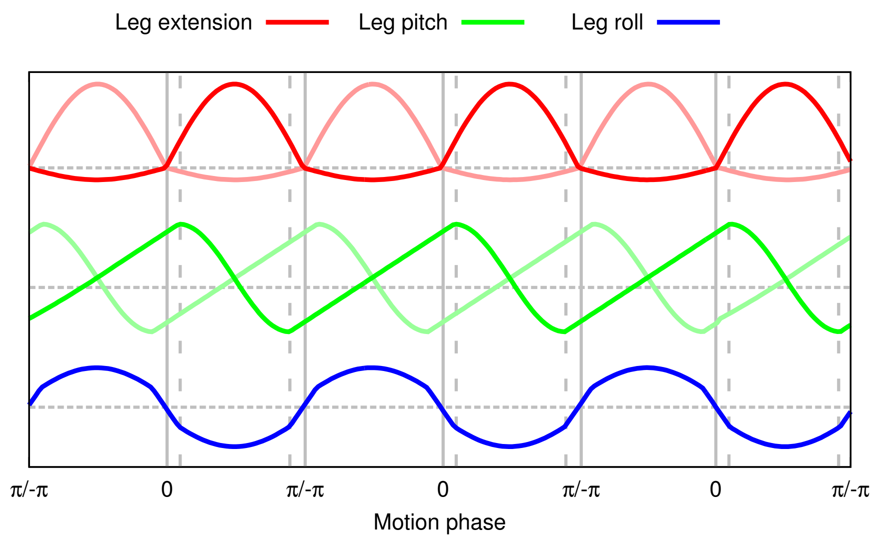


Figure 6.6: The main ingredients of the gait motion are rhythmical leg lifting (top), a leg swing motion (center), and a lateral hip swing (bottom). The solid vertical lines indicate the expected times of support exchange. The dashed vertical lines indicate the swing start and swing end timings. The patterns for the left leg (shown in faint color) are phase shifted by π.

6.3.1 *Halt Position*

The halt position is an offset from the Zero Pose that does not depend on the motion phase. It is the pose the robot stands in when the walking motion is not active. All other motion primitives are added to the halt position and thus it can be described as the "center" of the walking motion. Typically, in the halt position a robot has its knees slightly bent, the legs are spread apart by a few degrees to provide a wide enough stance, and the center of mass is adjusted to be roughly above the center of the feet. The halt position primitive

$$\mathbf{P}_{\mathrm{Halt}}(\lambda) = (P^{\eta}_{\mathrm{Halt}}, \mathbf{P}^{\mathrm{Leg}}_{\mathrm{Halt}}(\lambda), \mathbf{P}^{\mathrm{Foot}}_{\mathrm{Halt}}(\lambda)) \tag{6.10}$$

is a collection of configuration parameters that define the base values of the leg extension, leg angle, and foot angle parameters. The components of the halt position primitive are given by

$$P^{\eta}_{\mathrm{Halt}} = K_1, \tag{6.11}$$
$$\mathbf{P}^{\mathrm{Leg}}_{\mathrm{Halt}}(\lambda) = (\lambda K_2, K_3, 0), \tag{6.12}$$
$$\mathbf{P}^{\mathrm{Foot}}_{\mathrm{Halt}}(\lambda) = (\lambda K_4, K_5), \tag{6.13}$$

where $\lambda \in \{-1, 1\}$ denotes the leg sign.

The reason why the knees are typically not fully stretched is that when the leg extension is near zero, a small modification of the leg extension results in a relatively high required motor velocity. Some robots may also have a mechanical barrier that prevents the legs from being hyperextended. To avoid reaching the mechanical or velocity limits of the servo motors, the knees are slightly bent in the halt position. There is no numerical reason that prevents the legs from being fully stretched. The Leg Interface (6.2-6.9) is well-defined for a leg extension of $\eta = 0$.

6.3.2 *Leg Lifting*

Alternating leg lifting induces the lateral oscillation.

The leg lifting primitive is an alternating shortening of the legs that induces a lateral oscillation of the body mass and frees one leg at a time from its support duty. The leg lifting motion is shown as the topmost curve in Figure 6.6. Notably, the leg lifting primitive makes a distinction between a support phase, when the leg phase $\nu \leqslant 0$ and the foot is on the ground, and a swing phase, when the leg phase $\nu > 0$ and the foot is in the air. During the support phase, a small push is applied against the ground. During the swing phase, the foot is lifted up into the air and can be swung. The leg lifting primitive activates the leg extension parameter with a sinusoidal function

$$P_{\mathtt{LegLift}}(\nu, \mathbf{A}) = \begin{cases} \sin(\nu)\ \left(K_6 + K_7 \,\|\mathbf{A}\|_\infty\right), & \text{if } \nu \leqslant 0 \\ \sin(\nu)\ \left(K_8 + K_9 \,\|\mathbf{A}\|_\infty\right), & \text{otherwise} \end{cases} \tag{6.14}$$

that depends on the leg phase $\nu \in [-\pi, \pi)$ and the swing amplitude activation $\mathbf{A}$. The configuration variables K_6 and K_8 describe a constant push height during the support phase and step height during the the swing phase, respectively. In the support phase, the support leg pushes into the ground with a much smaller amplitude than it lifts up into the air during the swing phase. The push amplitude should never be greater than the leg extension in the halt position (K_1), otherwise the maximum leg extension will be reached and the leg cannot extend any further. The configuration variables K_7 and K_9 intensify the push and the step height depending on the L_∞ norm of the swing activation vector $\mathbf{A}$. The swing amplitude dependent increase of the step height (K_9) is crucial to avoid ground contact when large steps are taken and allows for calm steps with a low step height when the robot is walking slowly or in place. The support exchange is expected to occur at leg phases $\nu = 0$ and $\nu = \pm\pi$.

6.3.3 *Leg Swing*

The leg can be swung in the sagittal, lateral, and rotational directions.

To induce a walking motion in any direction, we use a leg swing pattern

$$\mathbf{P}_{\mathtt{LegSwing}}(\lambda, \nu, \mathbf{A}) = \left(P^{\mathtt{Roll}}_{\mathtt{LegSwing}}, P^{\mathtt{Pitch}}_{\mathtt{LegSwing}}, P^{\mathtt{Yaw}}_{\mathtt{LegSwing}}\right) \tag{6.15}$$

that activates the roll, pitch, and yaw components of the leg angle. The leg swing pattern is shown in Figure 6.6 in the center. The leg is swung forwards with a sinusoidal motion and pushed backwards with a linear motion during its support phase. The leg swing motion was designed this way with a specific motivation. If the legs were the spokes of a wheel that is traveling with a constant speed, the angular velocity of the spokes would be constant. The forward swing, however, is designed as a sinusoid in order to swing the leg as smoothly as possible, and to minimize inertial effects on the rest of the body.

The leg swing motion is not perfectly embedded into the leg phase. Swing phase configuration parameters K_{μ_0} and K_{μ_1} are used to delay the start of the swing motion and to rush the touchdown of the leg around the nominal support exchange time at leg phase $\nu = \pm\pi$. The parameters are indicated by dashed vertical lines in Figure 6.6. We found that these parameters are a good way of eliminating shuffling. Essentially, these parameters account for a short double support phase that occurs implicitly, even though it is not commanded by the leg lifting primitive.

To generate the leg swing pattern, we first compute a leg phase dependent unit swing oscillator

$$\zeta(\nu) = \begin{cases} \frac{2(\nu+2\pi-K_{\mu_1})}{2\pi-K_{\mu_1}+K_{\mu_0}} - 1, & \text{if } -\pi \leqslant \nu < K_{\mu_0} \\ \cos\left(\frac{\pi(\nu-K_{\mu_0})}{K_{\mu_1}-K_{\mu_0}}\right), & \text{if } K_{\mu_0} \leqslant \nu < K_{\mu_1} \\ \frac{2(\nu-K_{\mu_1})}{2\pi-K_{\mu_1}+K_{\mu_0}} - 1, & \text{if } K_{\mu_1} \leqslant \nu < \pi, \end{cases} \tag{6.16}$$

which incorporates the sinusoidal swing during the swing phase, the linear swing during the support phase, and the swing phase configuration parameters K_{ν_0} and K_{ν_1}. We use the swing activation vector $\mathbf{A}$ to modulate the amplitude of the unit swing oscillator in the roll, pitch, and yaw directions, and compute the leg angle parameter activators with the equations

$$P^{\mathrm{Roll}}_{\mathrm{LegSwing}}(\lambda, \nu, \mathbf{A}) = -\zeta(\nu)A_x K_{10} - \lambda \max(|A_x|K_{11}, |A_\psi|K_{12}), \tag{6.17}$$

$$P^{\mathrm{Pitch}}_{\mathrm{LegSwing}}(\lambda, \nu, \mathbf{A}) = \zeta(\nu)A_y K_{13}, \tag{6.18}$$

$$P^{\mathrm{Yaw}}_{\mathrm{LegSwing}}(\lambda, \nu, \mathbf{A}) = \zeta(\nu)\, A_\psi\, K_{14} - \lambda|A_\psi|K_{15}, \tag{6.19}$$

where $\lambda \in \{-1, 1\}$ denotes the leg sign.

Self-collisions must be avoided during sidestepping.

The leg swing equations (6.17-6.19) differ in the three directions. In the pitch direction, the legs are encouraged to swing fully from front to back. The maximum swing amplitude for forward walking and backward walking is configured using the step size parameter K_{13}. Laterally, however, the legs would collide. Therefore, leg roll angle offsets K_{11} and K_{12} are added proportionally to the roll and yaw swing amplitude activators A_x and A_ψ, causing the legs to spread out when walking in the lateral direction, and when the robot is turning.

In the yaw direction, an activation dependent yaw angle offset can be configured using the parameter K_{15}.

6.3.4 *Lateral Hip Swing*

The lateral hip swing pattern $P_{HipSwing}(\mu)$ activates the leg roll angle and sways the pelvis left and right during walking. It helps to transfer the weight from one leg to the other. The hip swing is illustrated in Figure 6.6 in the bottommost curve. In the case of the hip swing, both legs execute exactly the same pattern and, thus, the time argument of the pattern is the motion phase μ rather than the leg phase ν.

The hip swing is designed as a sinusoidal motion that includes transitions between swings to the right and swings to the left. The transition adjusts to the swing start timing K_{ν_0} and the swing stop timing K_{ν_1}. For the computation of the hip swing primitive, two explicit hip swing motion phases μ_l and μ_r are derived from the motion phase μ to determine the start and end points of two sine waves, one for the hip swing to the left, and one for the hip swing to the right, which overlap in the implicit double support phase. The sum of the two sine functions generates the final hip swing motion. μ_l and μ_r are given by

$$\mu_l(\mu) = \begin{cases} \mu - K_{\mu_1} + 2\pi, & \text{if } \mu < K_{\mu_0} \\ \mu - K_{\mu_1}, & \text{if } \mu > K_{\mu_1} \\ 0, & \text{otherwise,} \end{cases} \tag{6.20}$$

$$\mu_r(\mu) = \begin{cases} \mu - K_{\mu_1} + 3\pi, & \text{if } \mu + \pi < K_{\mu_0} \\ \mu - K_{\mu_1} + \pi, & \text{if } \mu + \pi > K_{\mu_1} \\ 0, & \text{otherwise.} \end{cases} \tag{6.21}$$

The hip swing to the left starts when the left foot touches the ground, and ends when the left foot is lifted off the ground. The hip swing to the right works in a symmetrical manner. The hip swing motion primitive is then computed with the equations

$$k = \frac{\pi}{K_{\mu_0} - K_{\mu_1} + 2\pi}, \tag{6.22}$$

$$P_{HipSwing}(\mu) = K_{16}\left(\sin(k\mu_l(\mu)) - \sin(k\mu_r(\mu))\right). \tag{6.23}$$

The two sine functions overlap and their sum creates a transition between the left and the right hip swing phases.

6.3.5 *Leaning*

The leaning motion primitive $\mathbf{P}_{Lean}(\mathbf{A}) = \left(P^{Roll}_{Lean}, P^{Pitch}_{Lean}\right)$ leans the robot slightly by adding an offset proportional to the swing amplitude to the roll and pitch angles of the legs. This primitive does not depend on the motion phase. The roll and pitch angle offsets are given by

$$P^{\mathrm{Roll}}_{\mathrm{Lean}}(\mathbf{A}) = -K_{17}A_{\psi}|A_y|, \tag{6.24}$$

$$P^{\mathrm{Pitch}}_{\mathrm{Lean}}(\mathbf{A}) = \begin{cases} K_{18}A_y, & \text{if } A_y \geqslant 0 \\ K_{19}A_y, & \text{if } A_y < 0. \end{cases} \tag{6.25}$$

The roll lean is determined as a value proportional to both the pitch and yaw leg swing amplitudes. Practically speaking, the robot "leans into curves". For the pitch lean, we distinguish between forward and backward walking, and use separate parameters to calibrate the lean offset for the pitch angle. In earlier work, we performed an experiment with the NimbRo-OP [Schwarz et al., 2012] that demonstrated that the pitch leaning primitive is of statistical benefit for the balance of the robot.

6.3.6 *Complete Leg Pattern*

The complete leg motion pattern

$$(\eta, \boldsymbol{\phi}_{\mathrm{Leg}}, \boldsymbol{\phi}_{\mathrm{Foot}}) = \mathcal{P}_{\mathrm{Leg}}(\lambda, \nu, \mu, \mathbf{A}) \tag{6.26}$$

computes a set of Leg Interface parameters for one leg. The parameters are computed by summing the appropriate motion primitives with the equations

The complete motion pattern is a sum of the motion primitives.

$$\eta = P^{\eta}_{\mathrm{Halt}} + P_{\mathrm{LegLift}}(\nu, \mathbf{A}), \tag{6.27}$$

$$\begin{aligned}\boldsymbol{\phi}_{\mathrm{Leg}} = \mathbf{P}^{\mathrm{Leg}}_{\mathrm{Halt}}(\lambda) + \big(P_{\mathrm{HipSwing}}(\mu), 0, 0\big) \\ + \mathbf{P}_{\mathrm{LegSwing}}(\lambda, \nu, \mathbf{A}) + (\mathbf{P}_{\mathrm{Lean}}(\mathbf{A}), 0)\,,\end{aligned} \tag{6.28}$$

$$\boldsymbol{\phi}_{\mathrm{Foot}} = \mathbf{P}^{\mathrm{Foot}}_{\mathrm{Halt}}(\lambda). \tag{6.29}$$

To obtain the Leg Interface parameters for both legs, we derive the time-shifted motion phase

$$\nu = \begin{cases} \mu + \pi, & \text{if } \mu + \pi < \pi \\ \mu - \pi, & \text{otherwise} \end{cases} \tag{6.30}$$

for use with the left leg. Then, we compute the Leg Interface parameters for the right leg using

$$\mathcal{P}_{\mathrm{Leg}}(1, \mu, \mu, \mathbf{A}) \tag{6.31}$$

and for the left leg using

$$\mathcal{P}_{\mathrm{Leg}}(-1, \nu, \mu, \mathbf{A}). \tag{6.32}$$

Please note the use of the appropriate leg sign and leg phase parameters. The flow of the motion phase μ and the value of the swing amplitude activation vector $\mathbf{A}$ are computed by the Control Interface, which is discussed in Section 6.4. At this point, it can be verified with the help of the leg swing motion pattern plotted in Figure 6.6, or the leg swing equations (6.17-6.19), that the complete motion pattern does indeed produce an approximately symmetrical step configuration with $\phi_{\mathtt{Leg}}^{\mathtt{Left}} \approx -\phi_{\mathtt{Leg}}^{\mathtt{Right}}$ in the moment of the support exchange. This is in accordance with our symmetrical step assumption in Chapter 5.

6.3.7 *Arm Motion*

The arm motion is generated in an analogous fashion to the leg motion with an arm motion pattern $\mathcal{P}_{\mathtt{Arm}}(\lambda, \mu, \mathbf{A})$. Similar to the Leg Interface, an Arm Interface provides an abstract actuator space in which halt and swing motion primitives are defined. The motion primitives are designed in the same way as for the legs. The arms swing antagonistically to the legs, i. e. the right arm swings forward when the left leg does, and the left arm swings forward when the right leg does. The most important role of the arm motion is to counteract the rotation about the support foot that would otherwise be induced by the inertia of the swinging leg.

6.3.8 *Compliant Actuation*

Simulated elastic actuation is achieved by configuring the position controller embedded in the Dynamixel servo motors of the robots with a low gain. All robots that the NimbRo motion pattern has been adapted to were actuated by servo motors of this brand. The soft feel of the low-gain position control leads to smooth and forgiving motions that dampen undesired vibrations during walking. The compliant actuation protects the transmission gears from mechanical damage to some extent and extends the operation time of the robots by using less energy and causing less heat in the motors.

Compliant actuation smooths the joint motions, but it also contributes to position tracking error and latency.

While the compliant motor setting results in smooth motions, it is important to be aware of the fact that it also increases the position tracking error of the motors. Figure 6.7 shows a time series of commanded and measured motor positions of the hip pitch and knee servos in a situation where the robot is walking forward. The position tracking error is quite evident. The low-gain position control has the same effect as a low-pass filter. It decreases the amplitude of the periodic motion output and introduces a phase shift, which contributes to the latency of the control loop. Within tolerable limits, the open-loop gait generator can compensate for the compliance, simply because the commanded motion signals can be configured to exag-

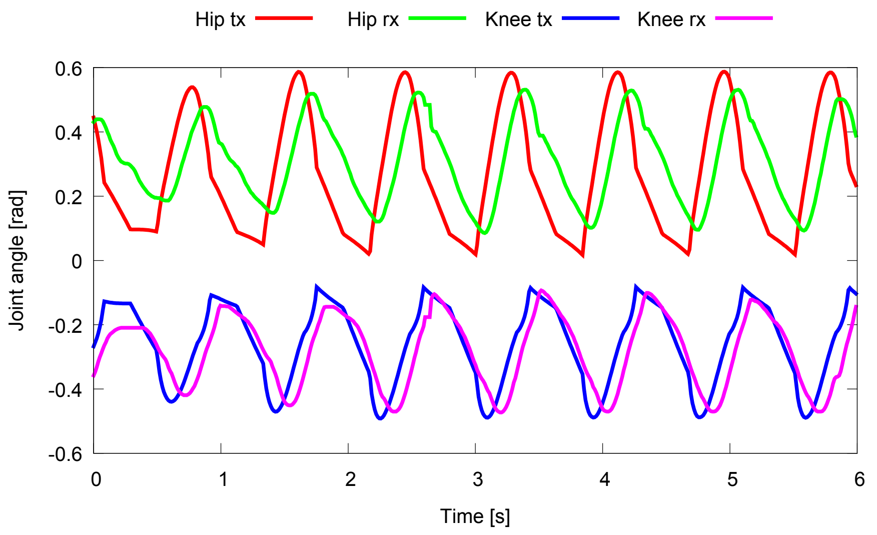


Figure 6.7: Time series of the commanded (tx) positions and measured (rx) positions of the right hip pitch and knee joints during forward walking, recorded with Dynaped. The compliant actuation mode smooths the motion output like a low-pass filter. It decreases the swing amplitude and introduces a phase shift.

gerate the leg swing motion in such way that the desired output of the motors is achieved. In a recent publication, Schwarz and Behnke [2013] proposed an approach to achieve precise position tracking with compliant motors, which is an option to address the position tracking problem on a more fundamental level.

Table 6.2: An annotated set of configuration parameters of the central pattern generator. The provided values are a set of parameters that were used for the bipedal robot Dynaped.

Variable	Value	Denotation
K_1	0.01	Halt Position leg extension
K_2	0.1	Halt Position leg roll angle
K_3	-0.06	Halt Position leg pitch angle
K_4	0.08	Halt Position foot roll angle
K_5	0	Halt Position foot pitch angle
K_6	0.01	Constant ground push
K_7	0	Proportional ground push
K_8	0.06	Constant step height
K_9	0.03	Proportional step height
K_{μ_0}	0	Swing start timing
K_{μ_1}	2.3876	Swing stop timing
K_{10}	0.12	Lateral swing amplitude
K_{11}	0.1	Lateral swing amplitude offset
K_{12}	0.01	Turning lateral swing amplitude offset
K_{13}	0.24	Sagittal swing amplitude
K_{14}	0.4	Rotational swing amplitude
K_{15}	0.05	Rotational swing amplitude offset
K_{16}	0.035	Lateral hip swing amplitude
K_{17}	0.07	Forward and turning lean
K_{18}	0.04	Forward lean
K_{19}	0	Backward lean
K_{20}	3.5	Swing amplitude limiting norm p
K_{21}	0.2	Lateral acceleration
K_{22}	0.2	Sagittal acceleration
K_{23}	0.2	Rotational acceleration
K_{24}	2.4	Constant step frequency
K_{25}	3.0	Maximum step frequency

6.4 CONTROL INTERFACE

The Control Interface exhibits parameters to control the size and timing of the steps.

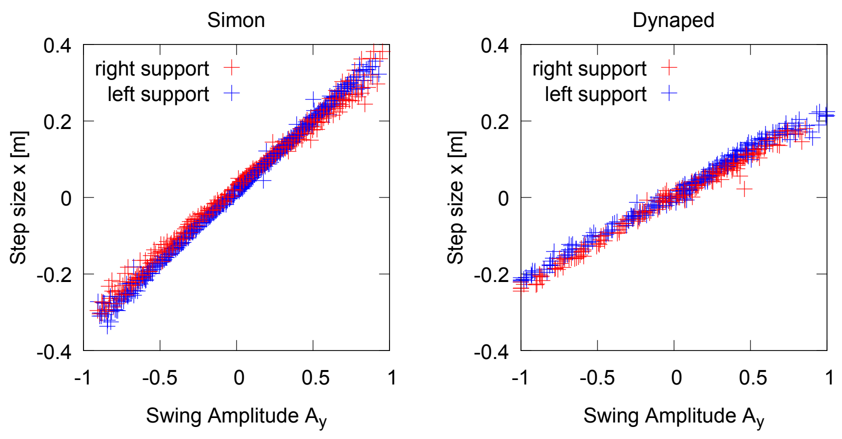
The top layer of the CPG chain is the Control Interface (recall Figure 6.2). The Control Interface exhibits two parameters that can be used to control the size and the timing of the steps. The desired size of the next step can be modified with the step size parameter $\check{\mathbf{S}} = (\check{S}_x, \check{S}_y, \check{S}_\psi)$, and the timing of the steps is set by the target step time $\check{T}$. The CPG will attempt to touch down the swing foot at the desired step coordinates $\check{\mathbf{S}}$ at time $\check{T}$, even if the parameters change mid-step, but the Control Interface will enforce a certain level of smoothness and bounds of feasibility. In this section, the conversion from desired step size $\check{\mathbf{S}}$ to swing amplitude activation $\mathbf{A}$, and from step time $\check{T}$ to motion phase μ, is explained.

6.4.1 *Step Size Conversion*

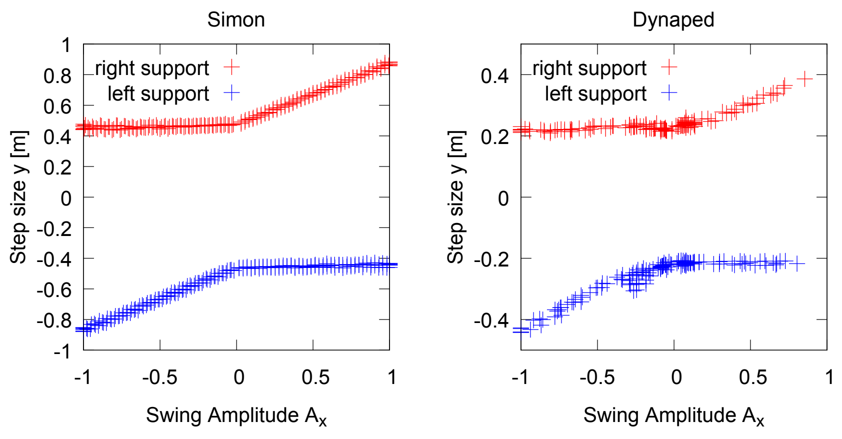
The swing amplitude vector $\mathbf{A} = (A_x, A_y, A_\psi) \in [-1, 1]^3$ contains activation signals for the leg swing in the roll, pitch, and yaw directions. It follows from the right-handed, x-forward coordinate frame assumption that the roll swing corresponds to lateral walking in the y direction of the robot, the pitch swing corresponds to forward and backward walking in the x direction, and the yaw swing corresponds to the rotational direction about the z axis. A swing activation value of 1 represents the highest achievable swing amplitude in the respective direction. For example, the swing activation vector $\mathbf{A} = (-0.8, 1, 0.5)$ means the robot should walk to the right with 80% of the possible step size, walk forward with the maximum possible swing amplitude, and turn left with half of the possible turning speed. Since we use the same leg swing amplitude $\mathbf{A}$ to generate the stepping motion for both legs, even though the phase is shifted for the left leg, the motion of the legs is coupled and one parameter per dimension is sufficient to control the gait. This is a key assumption that reduces the control space of the CPG to a low number of dimensions. Independent control of each leg would double the number of control parameters for the leg swing.

A linear relation between the leg swing amplitude and the resulting step size facilitates a simple conversion from desired step size to a leg swing activation signal.

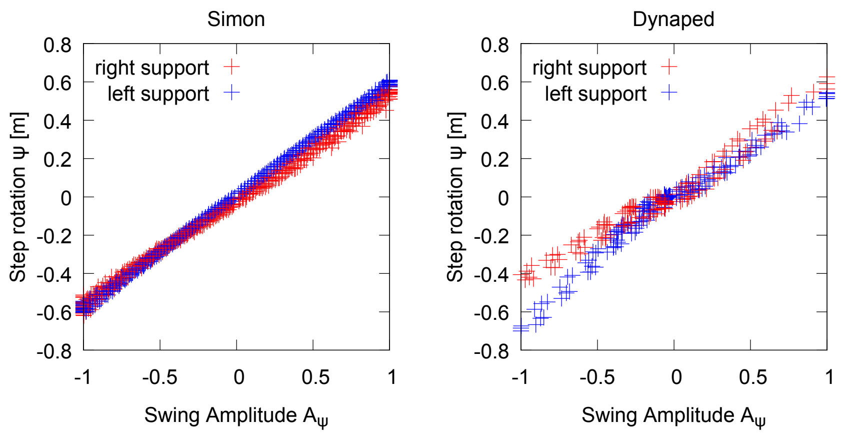
As previously described in Section 6.2, the equations of the Leg Interface (6.2-6.9) have been designed in such way that the leg angle parameter $\boldsymbol{\phi}_{Leg}$ is interpreted in a roll-pitch-yaw order. As illustrated in Figure 6.4a, this convention facilitates a one-to-one mapping of the leg roll angle to the y coordinate, the leg pitch angle to the x coordinate, and the leg yaw to the rotation of the footstep in the trunk frame, as long as only small angles are used to activate the leg roll and leg pitch swing. The roll and pitch leg swing equations (6.17, 6.18) together with the configuration parameter values K_{10}, K_{11}, and K_{13}, show that the commanded leg angle of Dynaped is always smaller than 0.24 radians. We assume furthermore, that the componentwise



(a) Sagittal



(b) Lateral



(c) Rotational

Figure 6.8: The step sizes in the (a) sagittal, (b) lateral, and (c) rotational directions can be approximated with linear functions of the swing amplitude parameter $\mathbf{A}$. In the lateral direction, two step sizes can be observed: a large step of variable step size by the leading leg, and a constant step size of the trailing leg. The data points are colored red when the right leg is the support leg, and blue when the left leg is the support leg. The left column shows data recorded with the simulated robot Simon. The right column shows real data from bipedal robot Dynaped. In the rotational direction the imperfection of the aging hardware is evident.

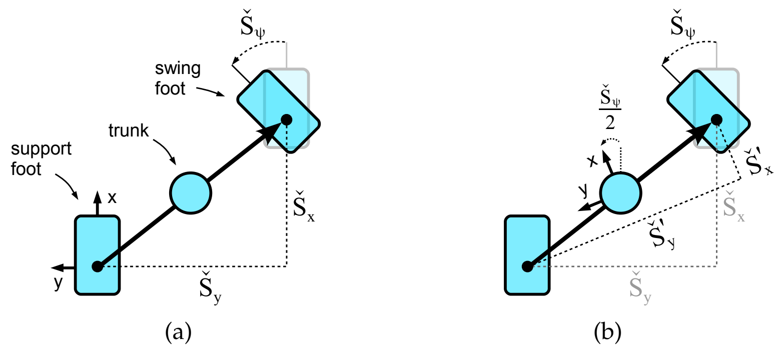


Figure 6.9: (a) The target step size $\check{\mathbf{S}}$ is expressed in the reference frame of the support foot. The rotational component $\check{S}_\psi$ defines the angle of rotation from the support foot to the swing foot. (b) To profit from a convenient linear conversion from step size to leg swing activation $\mathbf{A}$, the target step vector $\check{\mathbf{S}}$ has to be transformed to a step vector $\check{\mathbf{S}}'$ in the trunk reference frame.

mappings from leg swing amplitudes to step coordinates in the trunk frame are linear functions. Figure 6.8 confirms that our assumption is reasonable. The figure 6.8 shows plots of the foot-to-foot step size in trunk frame coordinates in each of the respective dimensions as a function of the leg swing amplitude. The step sizes were determined during walking by utilizing the kinematic model of the robot and measuring the distance from the support foot to the swing foot in the trunk frame at the moment of the swing foot touchdown. The state estimation procedure including the detection of the swing foot touchdown event and the measurement of the step size, is explained in Chapter 7. The data was collected with the simulated robot Simon and the real robot Dynaped. The maximal sagittal step size of 0.2 m for Dynaped multiplied by the open loop step frequency of 2.4 Hz yields a maximum open-loop walking speed of 0.48 m/s, sustainable only for short periods of time. When walking in the lateral direction, the biped takes a large step with the leading leg followed by a trailing step of smaller size. Thus, two different step sizes can be observed. The size of the leading step depends on the roll leg swing amplitude A_x, while the size of the trailing step does not.

The target step size $\check{\mathbf{S}} = (\check{S}_x, \check{S}_y, \check{S}_\psi)$ is a step size vector in Cartesian coordinates in the reference frame of the support foot, which is an x-forward, right-handed coordinate frame. The rotational component $\check{S}_\psi$ defines the angle of yaw rotation from the support foot to the swing foot. Figure 6.9a shows an example. To profit from the componentwise linear—and easily invertible—relation between the swing amplitude vector $\mathbf{A}$ and the step size in the trunk frame, the commanded step size must be transformed into the coordinate frame of the trunk. When performing a step, the CPG yaws both feet by half of the angle $\check{S}_\psi$ in opposing directions. By the end of the step, the trunk is rotated by half of the commanded angle $\check{S}_\psi$ with respect to the support foot. This is illustrated in Figure 6.9b. To transform the desired step size $\check{\mathbf{S}}$ to a desired step size $\check{\mathbf{S}}'$ in the trunk frame, we compute

$$\begin{bmatrix} \check{S}'_x \\ \check{S}'_y \end{bmatrix} = R(-\frac{\check{S}_\psi}{2}) \begin{bmatrix} \check{S}_x \\ \check{S}_y \end{bmatrix}, \tag{6.33}$$

$$\check{S}'_\psi = \check{S}_\psi, \tag{6.34}$$

where R is a counter-clockwise rotation matrix that negates the rotation of the trunk.

The leg swing activator is bounded and smoothed in order to protect the hardware from erratic motions.

Through the modulation of the motion pattern amplitudes in equations (6.14) and (6.17 - 6.19), the leg swing amplitude $\mathbf{A}$ has a direct effect on the motor targets that are sent to the robot. Therefore it is crucial that the Control Interface presents a swing amplitude vector to the Motion Pattern layer that remains continuous at all times in order to avoid sudden changes in the motor target positions that may lead to hardware damage. To this end, the desired step size $\check{\mathbf{S}}'$ in the trunk frame is mapped to an intermediate swing amplitude target $\check{\mathbf{A}} = (\check{A}_x, \check{A}_y, \check{A}_\psi)$ first, which is then smoothed and bounded before it is allowed to pass into the pattern generation layer. For the conversion of the desired step size $\check{\mathbf{S}}'$ in the trunk frame to the swing amplitude target $\check{\mathbf{A}}$, we exploit the linear correlation that we identified in Figure 6.9, and compute

$$\check{A}_y := k_x \check{S}'_x, \tag{6.35}$$

$$\check{A}_x := \begin{cases} \frac{\lambda(|\check{S}'_y| - S_y^{min})}{S_y^{max} - S_y^{min}}, & \text{if } \lambda\check{S}'_y - S_y^{min} > 0 \\ \check{A}_x, & \text{otherwise}, \end{cases} \tag{6.36}$$

$$\check{A}_\psi := k_\psi \check{S}_\psi, \tag{6.37}$$

where $\lambda = -sgn(\mu)$ is the sign of the support leg, determined as the opposite sign of the motion phase μ. The sagittal (6.35) and rotational (6.37) conversions are simple linear functions with factors k_x and k_ψ. For the lateral conversion (6.36), minimal and maximal step size parameters S_y^{min} and S_y^{max} are used, but only if the desired step is a leading step of at least the minimal size. Otherwise, the input is ignored and the roll swing amplitude target $\check{A}_x$ retains its previous value. The values of the conversion factors k_x and k_y, and the lateral step size boundaries S_y^{min} and S_y^{max}, are best determined experimentally from the data shown in Figure 6.8.

Leg swing amplitudes can be arbitrarily combined in all three dimensions as long as the swing amplitude is contained within a convex region defined by a p-norm. We regard p to be a parameter $K_{20} \geqslant 1$. The configurable shape of the applied norm restricts the target swing amplitude vector $\check{\mathbf{A}}$ to remain inside the region $[-1, 1]^3$, and also enforces a configurable restriction on the combination of the swing amplitude components. If after the step size conversion the K_{20}-norm of $\check{\mathbf{A}}$ exceeds a value of 1 ($\|\check{\mathbf{A}}\|_{K_{20}} > 1$), we normalize the swing amplitude target vector $\check{\mathbf{A}}$ using

$$\check{\mathbf{A}} := \frac{\check{\mathbf{A}}}{\|\check{\mathbf{A}}\|_{K_{20}}}. \tag{6.38}$$

The Control Interface maintains an internal state $\mathbf{A}$ of the swing amplitude that is moved towards the bounded target swing amplitude $\check{\mathbf{A}}$ in small increments within configurable limits

$$A_x := A_x + \min(\max(\check{A}_x - A_x, -K_{21}\rho), K_{21}\rho), \tag{6.39}$$

$$A_y := A_y + \min(\max(\check{A}_y - A_y, -K_{22}\rho), K_{22}\rho), \tag{6.40}$$

$$A_\psi := A_\psi + \min(\max(\check{A}_\psi - A_\psi, -K_{23}\rho), K_{23}\rho), \tag{6.41}$$

where $\rho = 0.01\,\mathrm{s}$ is the time period of one iteration of the main control loop (i. e. 100 Hz).

Both sensor noise and legitimate sudden changes of the task may cause the target step size $\check{\mathbf{S}}$ to change abruptly. As a result of the bounding and smoothing of the leg swing amplitude in the Control Interface, the target step size is allowed to change abruptly at any time during the step, and higher control layers are relieved of the responsibility of having to provide a continuous target. The robot will continue to execute a smooth stepping motion and attempt to step towards the target step location, to the extent that is possible given the remaining time of the step and the configured restrictions.

Please note that exposing the swing amplitude target $\check{\mathbf{A}}$ instead of the target step size $\check{\mathbf{S}}$ may be a more suitable interface for certain applications. For example, the output of a force field based gait control method, or direct joystick control by a human operator, can be more conveniently mapped directly to the unit region $[-1, 1]^3$ of the leg swing activation signal than to a commanded footstep location.

6.4.2 *Step Timing*

The step time parameter is converted into a step frequency, which is used to advance the motion phase.

The second parameter the Control Interface offers is the remaining time $\check{T}$ until the next support exchange. When the CPG walk is used in open-loop mode, a fixed frequency K_{20} pulses the step time parameter $\check{T}$. A closed-loop controller on a higher layer may adapt the step timing in order to maintain the balance of the robot. The technique used here is to convert the time-to-step parameter $\check{T}$ into a step frequency Ω. The step frequency determines by how much the motion phase μ is incremented per control loop iteration. We compute a step frequency such that the support exchange is induced exactly at time

$\check{T}$ seconds into the future. The time to frequency conversion is given by

$$\Omega = \begin{cases} -\frac{\mu}{\pi \check{T}}, & \text{if } \mu \leqslant 0 \\ \frac{\pi-\mu}{\pi \check{T}}, & \text{if } \mu > 0. \end{cases} \tag{6.42}$$

The motion generator reaches the support exchange at motion phases $\mu = 0$ and $\mu = \pm\pi$. Thus, two cases have to be handled, depending on the current value of the motion phase. The time to frequency conversion (6.42) becomes increasingly instable when the time-to-step $\check{T}$ approaches zero. The Control Interface bounds the step frequency to a range $[0, K_{25}]$. The applied bounds make sure that the step motion can only be played forwards, and that the increment of the motion phase is smaller than an admissible upper bound, preventing erratic step frequencies from generating discontinuous motor targets. Then, the motion phase μ is incremented based on the bounded step frequency

$$\mu := \begin{cases} \mu + \Omega\,\pi\,\rho, & \text{if } \mu + \Omega\,\pi\,\rho < \pi \\ -\pi & \text{otherwise,} \end{cases} \tag{6.43}$$

where $\rho = 0.01$ s is the time period of one iteration of the main control loop at 100 Hz. Whenever the motion phase exceeds the upper bound π, it needs to be reset to $-\pi$. This cannot be done smoothly. Thus, all phase dependent motion primitives have to be designed carefully in order to produce a smooth output even when the motion phase is reset.

6.5 EXPERIMENTS

6.5.1 *Video Demonstration*

Table 6.1 and the demonstration video [4] both show a number of different humanoid robots that the central pattern-generated walk described in this chapter has been used on. All of these robots could walk on a flat floor and demonstrated outstanding performance in RoboCup soccer games. In a push experiment that was performed in video [4], bipedal robot Dynaped was able to passively absorb impacts that were strong enough to create a visible disturbance during open-loop walking. The robot was also able to step on 2 cm thick board. Furthermore, the video shows the effects of compliant control. The robot automatically yields to pressure and dampens undesired swinging.

The NimbRo CPG has been successfully implemented on a number of bipedal robots of various sizes.

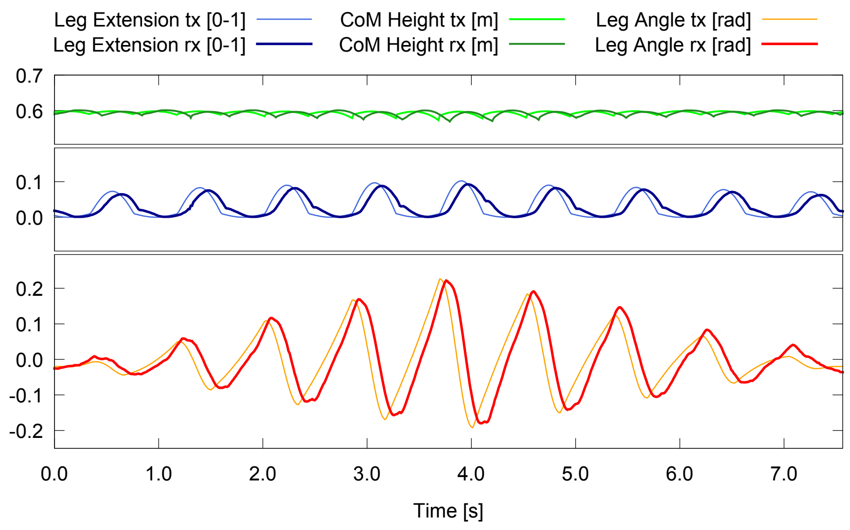


Figure 6.10: Center of Mass height, leg extension, and leg angle during walking. The Center of Mass travels on arc shaped trajectories rather than on a plane. The collision of the swing leg with the floor is automatically absorbed by the compliant actuators. Strong deviations between the commanded (tx) and the received (rx) leg extension motion signals can be observed shortly after the floor impact. The distance between the commanded and the measured leg angle gives a visual impression of the latency.

6.5.2 *Center of Mass and Compliant Actuation*

The center of mass travels on arcs and the floor impact is absorbed by the compliant actuators.

Figure 6.10 shows motion pattern data recorded with Dynaped while it was walking with the CPG. The experiment started with the robot walking in place. The robot accelerated to its peak velocity, slowed down again, and then came to a stop. The plot displays the CoM height on the top, and demonstrates the non-planar motion of the CoM. The data stream in the center shows the commanded signal (tx) for the leg extension and the signal received back from the robot (rx). Due to the compliant actuation, the swing leg automatically absorbs the impact of the floor contact. Accordingly, the received signal can be seen to deviate strongly from the commanded signal when shortly after the swing foot lands on the floor. The data stream in the bottommost plot in the figure shows the motion signal of the leg angle. There is an evident latency between the commanded signal and the received signal.

6.5.3 *Stability Analysis*

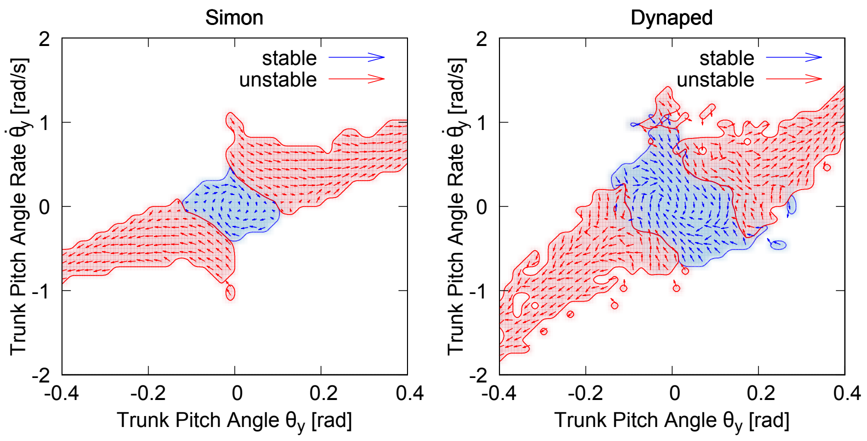
An open-loop stable basin of attraction is revealed in an exhaustive push experiment.

To analyze the open-loop stability of walking with the CPG, we performed push experiments with a simulated and a real robot. We commanded Simon and Dynaped to walk in place with the open-loop pattern generator. We subjected the robots to a large number of pushes of randomly varying strengths and made sure that strong enough pushes that make the robots fall were included. In two separate experiments, pushes were applied from the front and the back to examine the sagittal stability, and from the left and the right to examine the lateral stability. Simon was pushed 300 times in each experiment and fell between 100 to 120 times in each experiment. Dynaped was pushed at least 100 times in each experiment and fell approximately 60 times in each experiment. Please note that the number of falls does not reflect stability, as it can be easily influenced by the choice of the strength of the disturbance. The number of falls compared to the total number of pushes confirms that both types of pushes—those strong enough to make the robot fall and those weak enough to be passively absorbed—were included in sufficient numbers.

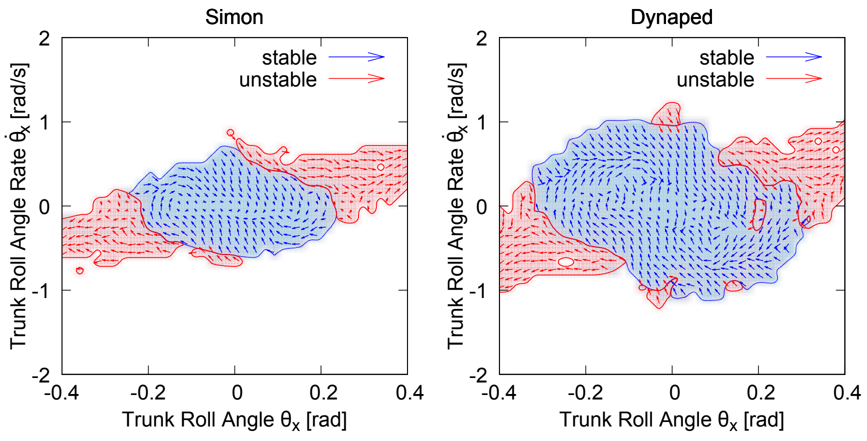
Figure 6.11 shows the results of this experiment. The plots show vector fields of stable and unstable areas in the pitch and roll phase spaces of the trunk angle. The sagittal stability is reflected by the pitch angle phase space, and the lateral stability is reflected by the roll angle phase space. The vector field is produced by constructing vectors that point from a trunk angle state to the trunk angle state 50 ms later. The vectors are then sorted into a grid of 40 by 40 cells and averaged to one vector per cell. Cells that do not contain at least two vectors before averaging are considered empty, and no vector is plotted. The length of the vectors has been capped to the radius of the in-circle of the cells in order to improve the visual effect. Short arrows symbolize a slow rate of change of the trunk angle. Long arrows correspond to a fast rate of change. The contours of the stable and unstable regions were determined by counting the visits in each grid cell. Dividing the number of visits that ended in a fall by the total number of visits yields the probability that the robot will fall after being in the state that is represented by the cell. Cells that have a higher than 50% probability of leading to a fall are marked in red. Cells that have a less than 50% probability of leading to a fall are marked in blue. Only cells that have been visited at least twice are assigned a color. The vector fields and the colored regions complement each other and reflect the response of the system to the applied push impulses. The arrows in the blue region point "inwards", leading the robot back to a stable state in the center of the phase space. The arrows in the red regions point "outwards", leading to an escalation of the trunk angle and a certain fall.

The existence of a stable region is best explained by the non-zero size of the feet, and by the width of the stance in the lateral direction. When the robot is undisturbed, the pitch and roll angles and angular velocities are near zero, i. e. in the center of the blue regions. Pushes tilt the robot and drive the trunk angle away from the center position. If the push was weak enough that the robot does not tip over the edge of a foot, the trunk angle and angular velocity remain within the blue colored basin of attraction and spiral back to the center of the phase

The set of stable states near the upright pose is a result of the non-zero size of the feet.



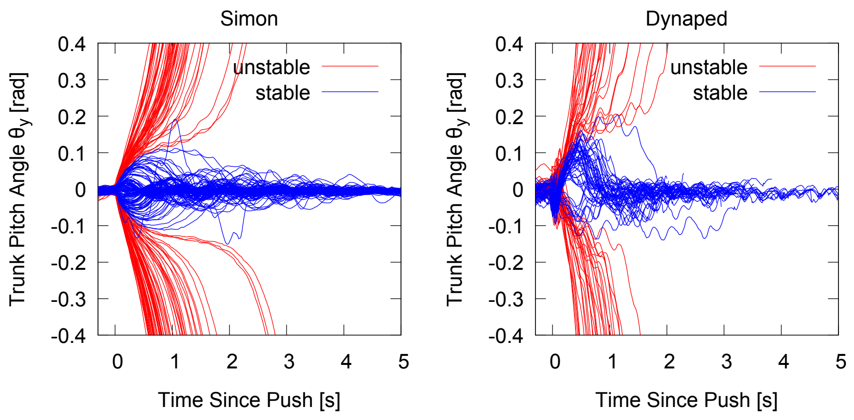
(a) Sagittal



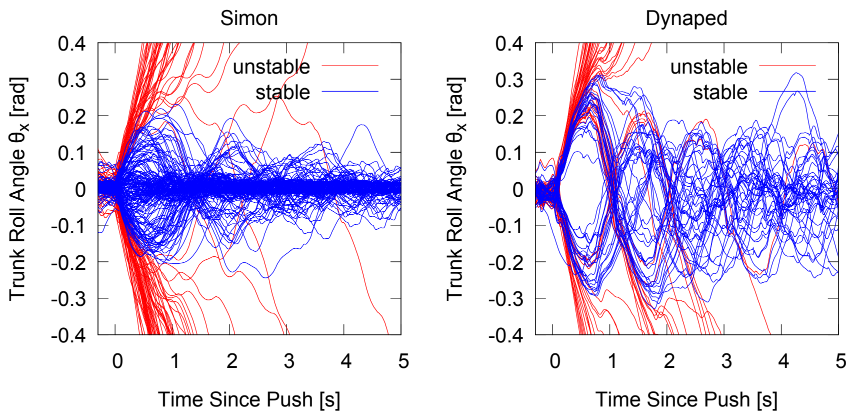
(b) Lateral

Figure 6.11: Vector fields of the pitch and roll trunk angle phase spaces. The simulated robot Simon and the real robot Dynaped were subjected to pushes of randomly varying strength from the front and the back to explore the (a) sagittal (pitch) stability, and from the left and the right to examine the (b) lateral (roll) stability. The vectors hint at the short-term future development of the trunk angle. The length of the arrows reflects the angular velocity up to an upper bound for better visibility. For each plot, a stable region is outlined in blue. States that were frequently involved in a fall are marked in red. Both the pitch and roll phase spaces possess a stable region where the robot is able to passively absorb disturbances. The size of the stable regions is mostly to be attributed to the size of the feet in the sagittal direction, and to the width of the stance in the lateral direction.

space. If the push is strong enough to tip the robot over the edge of a foot, it drives the trunk angle state into one of the red regions and the robot inevitably falls. Despite fewer and more noisy samples from the real robot, the same regions can be identified in the trunk angle phase spaces. Interestingly, larger stable regions appear in both trunk angle phase spaces of Dynaped. The reason for this must be the larger foot size to CoM height ratio of Dynaped compared to Simon.



(a) Sagittal



(b) Lateral

Figure 6.12: Plots of (a) pitch and (b) roll trunk angle trajectories synchronized at the moment of the push.

Generated from the same push experiments, Figure 6.12 shows plots of the trunk pitch angle and the trunk roll angle after the push, synchronized by the push event. In the same manner as before, trajectories that resulted in a fall are marked in red. Stable trajectories are marked in blue. A noticeable threshold separates the trajectories that were able to return from the edge of the foot from the ones that tipped over and ended in a fall. In the lateral experiment, once the oscillation has been disturbed, it takes a long time for it to settle and to return to its usual amplitude. Some of the falling trajectories were not brought to a fall directly by the push, but oscillate first before tipping over. These cases present evidence that the robot can disturb itself by pushing into the ground with a badly timed step.

6.6 DISCUSSION

The presented CPG creates an open-loop walking motion for bipedal robots using configurable motion patterns that operate in the parameter space of an abstract motion interface. It combines well with elastic, low-gain position control. The central pattern generator is model free in that it uses an abstract kinematic interface that does not relate end-effector configurations to joint angles. Masses, inertia, and sizes of body parts do not need to be known. A successful history of walking and soccer playing bipedal robots provides evidence that self-stable walking can be achieved with this CPG.

The Leg Interface—the kinematic abstraction layer the CPG is based on—constitutes the encoding of intuitive leg motion components that are used during walking. Using the Leg Interface can produce a natural looking and energy-efficient robotic walk with stretched knees. An inverse kinematics-based interface does not naturally support stepping motions with stretched knees. When using inverse kinematics to track motion trajectories defined in Cartesian space, end-effector targets must be designed in such way that they remain inside the kinematically feasible region of the legs. When using the Leg Interface to express motions, this is always guaranteed. Due to the encoding of motions using angles, the motion patterns can be transferred between robots of different sizes without the need for scaling or retargeting. The necessity to learn the conversion from footstep parameters to leg swing amplitudes, which we accomplished using simple linear mappings, appears to be a disadvantage of the abstract kinematic interface, as compared to inverse kinematics. However, even in the latter case, where the transformation of the footstep parameters to joint angles is given directly, elastic actuator elements and other sources of imprecise actuation can cause unexpected deviations from the computed results and may necessitate a similar learned mapping.

Due to the execution of the same pattern with both legs, the Control Interface can offer a conveniently low number of parameters to control the walking motion on a higher level of abstraction. Both the computation of an analytic balance controller and the online learning of balance profit from this simplification.

We argue that it is beneficial to use an open-loop stable algorithm as a building block in a hierarchical gait control approach. Most importantly, a reference trajectory can be gained by modeling the periodic motion of the center of mass observed during open-loop walking. This reference trajectory is a product of the natural dynamics of the physical system and represents a stable limit cycle. The reference trajectory can then be used as a target for a light-weight control strategy that simply attempts to return to a motion that the robot is comfortable with. Most of the successful walking algorithms use some sort of a reference trajectory [Englsberger et al., 2011, Kajita et al., 2003, Urata et al., 2012]. However, they are generated by a low-dimensional model and not by a self-stable whole-body walking motion. Furthermore, in situations where the robot is stable and corrective actions are not required, the balance control loop can remain inactive and allow the robot to walk without control effort.

A drawback of the central pattern generator is the need for manual parameter tuning. We envision future motion generators as being able to tune their own parameters during operation times, optimizing the cost of transportation and slowly adjusting to changes of the body over time.

Throughout the remainder of this thesis, we can regard the pattern generator as a black box that can be commanded to step onto Cartesian floor coordinates at specific times. We now begin to build a balance control augmentation on top of the CPG that uses the Control Interface to command the pattern generator to step onto desired locations. In the next chapter, we introduce our method to compute a low-dimensional feature that represents the state of balance and serves as a basis for the computation of the step size and the step timing.

7 STATE ESTIMATION

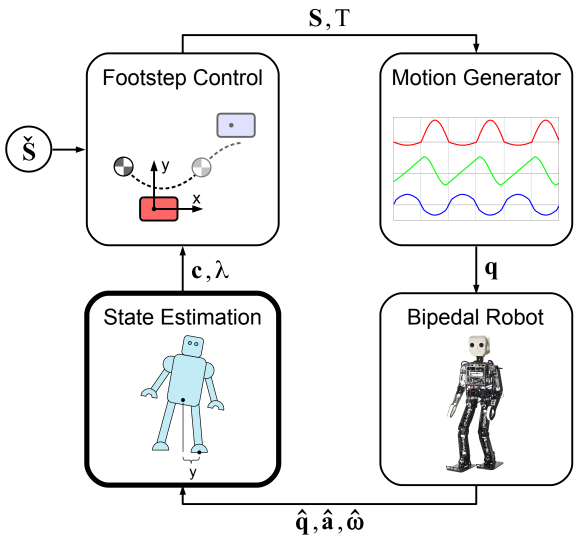


Figure 7.1: The State Estimation module uses the joint angles $\hat{\mathbf{q}}$ measured by motor encoders, the inertial acceleration $\hat{\mathbf{a}}$ of the trunk, the angular velocity $\hat{\boldsymbol{\omega}}$ of the trunk, and a kinematic model to reconstruct the tilted whole-body pose of the robot. For the purpose of balance control, the position and the velocity of the Center of Mass $\mathbf{c}$ and the support foot sign λ are extracted from the pose reconstruction.

The state of the Center of Mass and the support leg indicator are the main features computed by the State Estimation.

The State Estimation module illustrated in Figure 7.1, reconstructs the whole-body pose of the robot in three-dimensional space. To this end, it uses a kinematic model and the sensor information gained from the robot. The sensor inputs into the State Estimation module are the joint angles $\hat{\mathbf{q}}$ measured by motor encoders, the inertial acceleration $\hat{\mathbf{a}}$ of the trunk measured by accelerometers, and the angular velocity $\hat{\boldsymbol{\omega}}$ of the trunk reported by gyroscopes. For the purpose of generating a stable walk, we are interested in two particular features: the coordinates and the velocity of a fixed point on the body frame, which we refer to as the Center of Mass (CoM) state $\mathbf{c}$, and an indicator $\lambda \in [-1, 1]$ for the sign of the leg the robot is currently standing on. Both of these features can be extracted from the whole-body pose reconstruction.

In the following, we describe a common method of fusing the accelerometer and the gyroscope data into an estimate of the trunk attitude $\boldsymbol{\theta}$, which is required for the pose reconstruction. Then, we explain our method of tilted whole-body pose reconstruction, CoM state extraction, and support leg sign detection.

7.1 TRUNK ATTITUDE ESTIMATION

The measurements of accelerometers and gyroscopes are fused into a trunk attitude estimation, exploiting the advantages of both types of sensors.

An Inertial Measurement Unit (IMU), typically mounted in the trunk of a humanoid, uses a combination of accelerometers and gyroscopes, sometimes also magnetometers, for the purpose of estimating the orientation of a body. Accelerometers report the vector of the total inertial acceleration experienced by the sensor. A dominant part of the acceleration is due to the gravitational force, which allows a raw estimate of the trunk attitude to be computed directly from the accelerometer readings. The raw estimate is not always accurate, however, because the accelerometer readings also contain the acceleration resulting from other forces at work on the body. Moreover, the trunk of a walking robot is shaky, and thus the accelerometer readings are quite noisy. Gyroscopes report the angular velocity in a body-fixed coordinate frame, which is much less prone to noise. The angular velocity can be integrated over time to obtain a low noise angle estimate. This suffers from drift, because the integration accumulates errors. It is common practice to combine the advantages of both sensors to form a low-noise, drift-free, and low-latency estimation of the attitude with the use of a complementary filter.

The trunk attitude $\boldsymbol{\theta} = (\theta_x, \theta_y)$ is expressed as a trunk roll angle θ_x and a trunk pitch angle θ_y about the x and y axes of a body-fixed right-handed coordinate frame, respectively, where the x-axis points forward, and the y-axis points to the left. The trunk attitude is interpreted as a vector of independent, one-dimensional rotations with respect to the world vertical. While this definition does not hold for pitch and roll combinations that considerably deviate from vertical, it is sufficient for the purpose of presenting the concepts of this thesis, and for balance control in a small range of trunk attitudes around the upright pose. A more robust definition of "fused angles" suitable for use in more general 3D scenarios has been published by Allgeuer and Behnke [2014a].

Let $\hat{\boldsymbol{a}} = (\hat{a}_x, \hat{a}_y)$ be a two-dimensional vector of inertial acceleration measured by accelerometers with its components expressed in units of the gravitational constant G and bounded to $[-1, 1]$. The raw trunk angle estimate $\hat{\boldsymbol{\theta}} = (\hat{\theta}_x, \hat{\theta}_y)$ is then given by

$$\hat{\theta}_x = \arcsin(\hat{a}_x), \tag{7.1}$$

$$\hat{\theta}_y = \arcsin(\hat{a}_y). \tag{7.2}$$

Let $\hat{\boldsymbol{\theta}}_n$ be the raw trunk angle estimate, and $\hat{\boldsymbol{\omega}}_n$ be the angular velocity reported by the gyroscopes in iteration n of the control loop. Then, the trunk angle estimate for that iteration, $\boldsymbol{\theta}_n$, is given by

$$\boldsymbol{\theta}_n = (1-k)\left(\boldsymbol{\theta}_{n-1} + \rho\left(\hat{\boldsymbol{\omega}}_n - \mathbf{B}_n\right)\right) + k\hat{\boldsymbol{\theta}}_n, \tag{7.3}$$

where $k = 0.01$ is a blending factor, $\rho = 0.01\,\mathrm{s}$ is the time interval between iteration n and $n-1$, and $\mathbf{B}_n$ is the gyro bias. The idea behind this formula is that the blending of the lowly weighted raw trunk angle angle with the highly weighted gyroscope integration "ties" the low-noise gyro integration to the noisy but drift-less signal of the accelerometers. The gyro bias $\mathbf{B}_n$ plays an important role for the accuracy of the attitude estimation. Typically, the gyroscope output is prone to systematic error, e. g. a constant overestimation of the angular velocity. The integration of the raw gyroscope output accumulates this systematic error and results in an attitude estimate that drifts away from the real angle with time. Subtracting the bias before performing the integration is a good way of sufficiently reducing the drift. One can estimate the gyro bias in a simple way by computing the mean of the angular velocities that the sensor reports in a state of rest. A better method, however, is to estimate the bias during operation by utilizing the angle estimate provided by the accelerometers. This can be done using

$$\bar{\theta}_n = \bar{\theta}_{n-1} + \rho\hat{\omega}_n, \tag{7.4}$$

$$\beta_n = \frac{(\bar{\theta}_n - \bar{\theta}_{n-m}) - (\hat{\theta}_n - \hat{\theta}_{n-m})}{\rho m}, \tag{7.5}$$

$$\mathbf{B}_n = \mathbf{B}_{n-1} + k(\beta_n - \mathbf{B}_{n-1}). \tag{7.6}$$

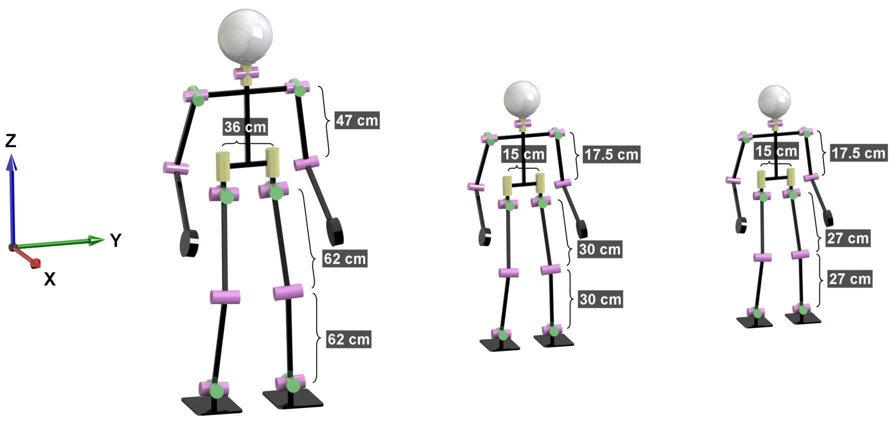
In (7.4), we integrate the raw gyroscope readings $\hat{\omega}$ into a biased angle estimate $\bar{\theta}$ and compare by how much the estimated angle changed during the last m iterations according to the biased integration $\bar{\theta}$ and according to the accelerometer readings $\hat{\theta}$ in (7.5). Their difference divided by the time of m iterations yields a new gyro bias estimate β_n, which we slowly blend our running bias estimate $\mathbf{B}_n$ towards with a blending factor of $k = 0.001$. Aside from the automatic calibration of the gyro bias, the advantage of this method is that it can account for a slow change of the bias during runtime.

A more sophisticated trunk attitude estimator that performs the estimation in 3D and also includes a magnetometer sensor has been published recently by Allgeuer and Behnke [2014b].

7.2 TILTED WHOLE-BODY POSE RECONSTRUCTION

A tilted whole-body pose reconstruction with the help of a robot skeleton is a rich source of kinematic and balance-related information.

We use the joint angle information $\hat{q}$ obtained from the motor encoders and the estimated trunk attitude θ to reconstruct a tilted whole-body pose of the robot with the help of its kinematic model. The measured joint angles $\hat{q}$ are applied to set the kinematic model in pose with a forward kinematics algorithm. After this operation, the relative orientation of the rigid body parts with respect to their parent in the kinematic chain matches the orientation implied by the set of joint angles that were measured from the robot. Once in pose, the entire kinematic model is rotated around the center of the current support foot such that the trunk attitude equals the roll and pitch



(a) Simon (b) Copedo (c) Dynaped

Figure 7.2: The kinematic models used for (a) the simulated robot Simon and smaller models used for the soccer robots (b) Copedo and (c) Dynaped. Sizes are not to scale.

angles $\theta = (\theta_x, \theta_y)$ returned by the complementary filter. We consciously neglected the fact that the robot would rotate about one of the edges of the support foot, and not about the center of the foot. This eliminates the need to determine the edge to rotate about, and disposes of a potential source for jitter at the cost of a negligible error.

In this manner, we aggregate a pose reconstruction that combines the kinematic state of the robot with its state of balance. Since pitch and roll estimates were incorporated, the position and orientation of body parts relative to the floor can be determined, under the assumption of a flat and horizontal floor. In this way, the pose reconstruction is a useful source of balance relevant information, such as for example the angle of the support foot relative to the floor, the position and orientation of the trunk relative to the floor, and the position and orientation of the swing foot relative to the floor. The global yaw orientation is not included in our model, as it is not relevant for balance. Including the yaw orientation, however, could extend the usefulness of the tilted whole-body pose reconstruction for localization purposes.

Omitting small details from the kinematic model assists mathematical operations.

Without attempting to be precise, we use a generic humanoid kinematic model that we developed in simulation [Missura and Behnke, 2013b] and adjust the lengths of its body segments to the sizes of the controlled robot. The layout of the kinematic model and the sizes of the relevant segments are illustrated in Figure 7.2. In this generic humanoid model, the rotational axes of all degrees of freedom of a joint intersect at a single point. This is not always the case with real hardware. The neglecting of the small offsets between the rotational axes simplifies the implementation of the forward and inverse kinematic operations, the adaptation to different robot prototypes, and accelerates physical simulation.

7.3 CENTER OF MASS STATE ESTIMATION

Using the reconstructed pose in three-dimensional space, we track the motion of the center point in between the hip joints with respect to a footstep frame. The footstep frame is set to the ground projection of the new support foot in the moment of a detected support exchange. Support exchange detection is discussed in Section 7.4. The ground projected support frame is horizontally aligned with the floor and preserves the yaw orientation the new support foot had relative to the last footstep frame after the support exchange. During the step, the footstep frame remains fixed until the next support exchange occurs. With respect to the footstep frame, we compute the coordinates of the ground projected center point between the hip joints and obtain the CoM state $\mathbf{c} = (c_x, \dot{c}_x, c_y, \dot{c}_y)$, a four dimensional vector with position and velocity components in the sagittal and lateral directions. The values of the derivatives $\dot{c}_x$ and $\dot{c}_y$ have to be determined by numerical differentiation and are hence prone to noise.

The center point between the hip joints can be used as an approximation of the center of mass.

The relocation of the footstep frame in the moment of the support exchange introduces an unavoidable discontinuity in the CoM trajectory. To maintain a continuous velocity in the global reference frame, the velocity vector $(\dot{c}_x, \dot{c}_y)$ is rotated into the new support frame in the event of a support exchange, using

$$\begin{bmatrix} \dot{c}_x \\ \dot{c}_y \end{bmatrix} := R(-\phi) \begin{bmatrix} \dot{c}_x \\ \dot{c}_y \end{bmatrix}, \tag{7.7}$$

where $R(-\phi)$ is the 2D rotation matrix corresponding to a counter-clockwise rotation of $-\phi$, and ϕ is the yaw angle that rotates the footstep frame before the support exchange to the footstep frame after the support exchange. The magnitude of the velocity vector is not updated until at least two CoM state estimates are available in the new support frame to continue the numerical differentiation.

7.4 SUPPORT FOOT ESTIMATION

The support foot estimation is a continuous process and can be initialized with either the right or the left foot. If after the pose reconstruction outlined above, the vertical coordinate of the swing foot has a value lower than the vertical coordinate of the support foot, the roles of the feet are switched and the sign $\lambda \in \{-1, 1\}$ of the support foot is set to either to $\lambda = -1$ for the left foot, or $\lambda = 1$ for the right foot. In this moment the support frame is relocated to the ground projection of the new support foot. In order to avoid erratic changes of the support foot sign when both feet are on the ground, after every change of the support role, we require the vertical distance between the feet to exceed 5 mm before another support exchange is allowed to occur.

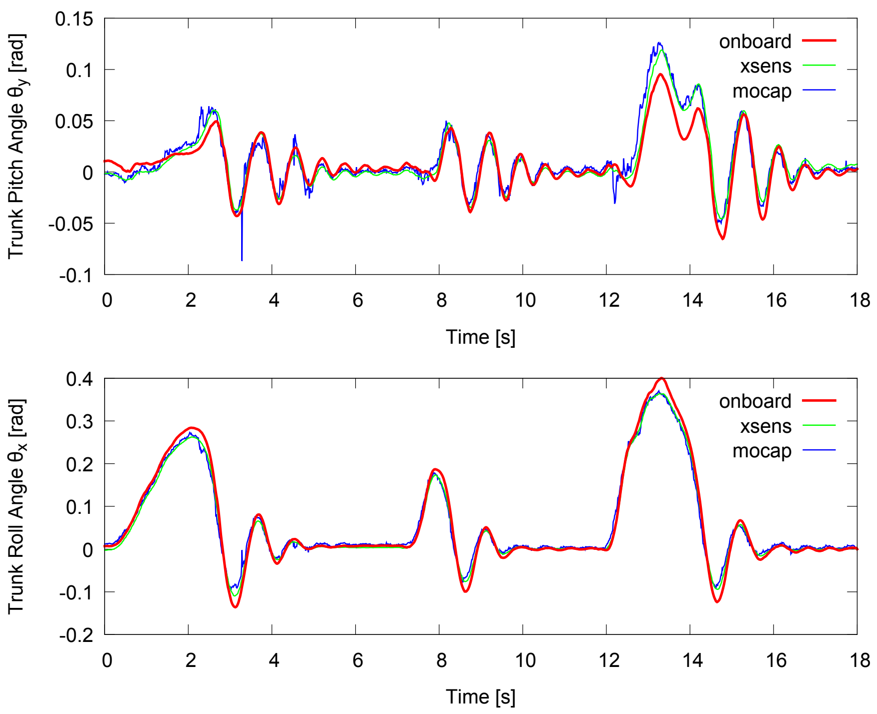


Figure 7.3: Comparison of the attitude estimation with the onboard equipment of robot Copedo, a motion capture device, and a commercial Xsens sensor. In the pitch and roll directions, the robot was slowly pushed onto a support edge and released shortly before tipping over, allowing the robot to rock back into a standing position.

Note that this support foot detection method is based on the assumption that the floor is horizontal and flat, and at least one foot touches the ground at all times. Based on these assumptions, support foot detection is possible without foot pressure or ankle torque sensors, but the method does not scale to non-planar surfaces.

7.5 EXPERIMENTS

7.5.1 *Trunk Attitude Estimation*

We evaluated the quality of our trunk attitude estimation in a motion capture experiment. Bipedal robot Copedo was equipped with reflective markers and their position in three-dimensional space was determined by an OptiTrack[1] motion capture device. From multiple markers attached to the trunk, it is possible to reconstruct the trunk orientation. We also attached a commercially available Xsens sensor to the trunk. This sensor has an integrated filter and reports a direct trunk attitude estimate. In separate experiments for the pitch and roll directions, Copedo was pushed onto a support edge and released

1 http://www.naturalpoint.com/optitrack

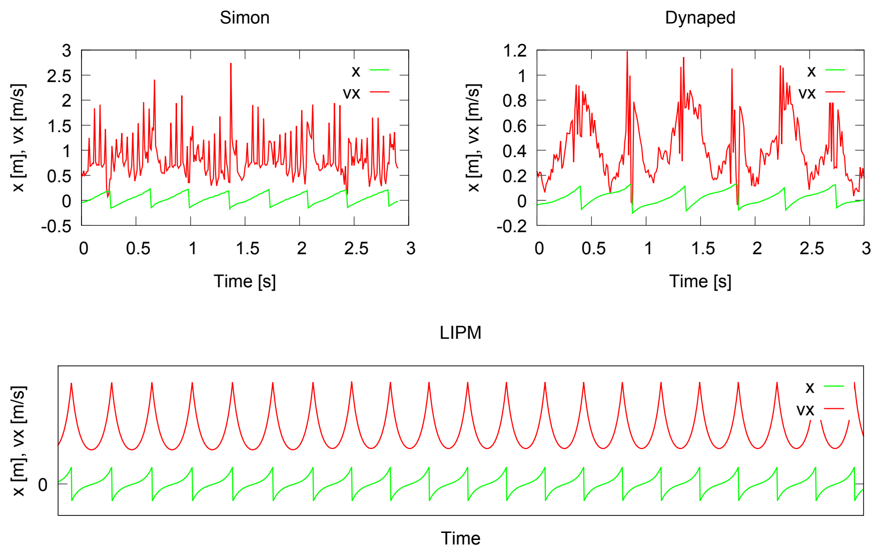
shortly before tipping over, allowing it to rock back into a standing position. This was repeated several times in each direction. Figure 7.3 shows a comparison of the trunk pitch and trunk roll angles estimated with the onboard sensor equipment of Copedo (filtered with the complementary filter), the motion capture device, and the Xsens sensor. Especially at large peaks, the motion capture device and the Xsens sensor are more in agreement with each other, than with our attitude estimation. The motion capture data is relatively noisy, but it is certain to be drift-free. The Xsens sensor is a quality product of an established brand that appears to perform reliably. Based on the agreement of the two other sources of data, the self-assembled sensor equipment of robot Copedo appears to show some deficiency. It is unclear how much of the deviation can be attributed to the hardware components and the simple complementary filter. However, the accuracy is sufficient for balanced walking and push recovery.

7.5.2 *Center of Mass State Estimation*

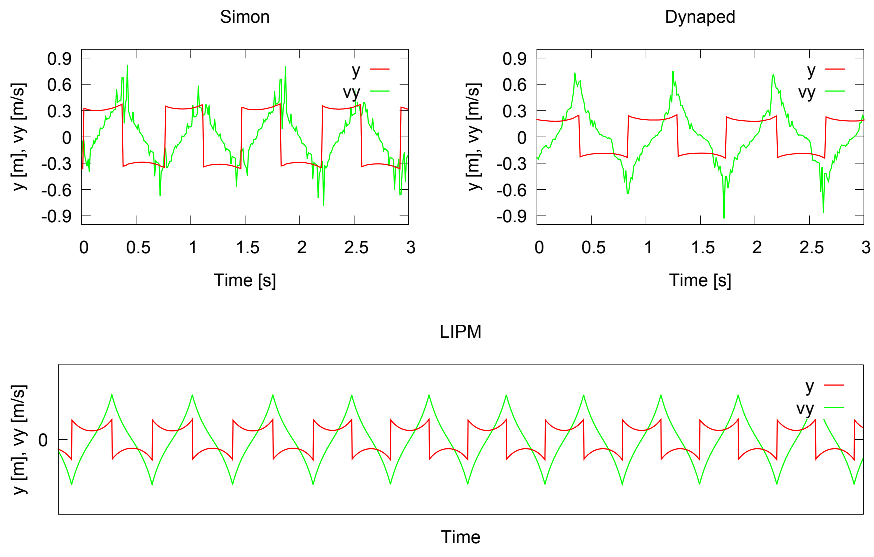
Figure 7.4 shows CoM positions and velocities in the sagittal and lateral directions for the simulated robot Simon and the real robot Dynaped. The CoM positions and velocities were estimated with the tilted kinematic pose reconstruction method that was introduced earlier in this chapter. The data was recorded during an open-loop experiment where the robots Simon and Dynaped were walking with the CPG. The data also shows the support exchange detection where discontinuities in the CoM position trajectory occur. It is obvious that the velocity estimates are rather noisy in comparison to the position estimates. This is due to the fact that the velocity of the CoM is determined by means of numerical differentiation. Interestingly, the velocity estimation noise in the simulated environment is significantly larger than on the real robot. The reason for this is that the motion of rigid bodies within the Bullet Physics engine is jerky on a small time scale, since Bullet uses a combination of actually simulated world state updates and interpolations in between states to speed up performance. Whenever a real simulation step occurs, the position and orientation of the involved rigid bodies is incremented in a different way, resulting in a spike in the velocity estimate. Increasing the simulation frequency results in less noise, but is more computationally intense. The Capture Step controller includes a predictive filter that is able to reduce the noise we experience in simulation. The filter is presented in Section 8.2. The most characteristic sources of noise in the real robot data are the support exchanges, where spikes in the velocity estimate can be observed. The predictive filter includes a step noise suppression mechanism to solve this issue.

The velocity of the center of mass cannot be determined as precisely as its position.

Most importantly, the CoM data of the robots resembles the motion of an inverted pendulum in both the sagittal and lateral directions.



(a) Sagittal motion



(b) Lateral motion

Figure 7.4: Estimated Center of Mass position and velocity data in the sagittal and lateral directions for the simulated robot Simon and the real robot Dynaped. The velocity estimates are noisy because they are determined by means of numerical differentiation. Interestingly, the velocity estimates in the physical simulation are noisier than on the real robot. Example Linear Inverted Pendulum Model trajectories are shown in the elongated plots. When walking with the open-loop Central Pattern Generator, the Center of Mass motion of the robots in the (a) sagittal and (b) lateral directions resemble the motion of an inverted pendulum.

The elongated plots in Figure 7.4a and 7.4b show schematic trajectories generated with the LIPM. The Linear Inverted Pendulum Model introduced in Chapter 8.1 is a mathematical concept that approximates the dynamics of an inverted pendulum in closed form. The shapes of the LIPM position and velocity profiles are similar to the position and velocity profiles produced by the robots. This is particularly interesting because the walking motion of the robots was generated by the CPG, which does not explicitly attempt to generate an inverted pendulum-like walk.

The Center of Mass motion gained from a simulated and a real robot resembles the motion of an inverted pendulum.

7.6 DISCUSSION

The low-dimensional representation of the motion state is an essential feature of the Capture Step Framework. The fast, analytic computation of suitable footstep control equations is based on the high correlation between the motion of the extracted CoM state **c** and a motion trajectory engineered with the Linear Inverted Pendulum Model. Even though we compress the whole-body state into a planar model, rotations of the whole body about the support foot are still accounted for since we rotate the kinematic model by the trunk attitude before extracting the CoM state. We do not assume that the feet remain flat on the floor. This is a strong distinction between our work and the majority of the prevailing state of the art approaches. Push experiments we perform in subsequent chapters show that strong disturbances inevitably lead to a violation of the floor-aligned support foot assumption. Nevertheless, the balance controller is able to recover from oblique poses to a certain degree, despite only using a planar model.

The planar model for state representation still accounts for rotations about the support foot.

By tracking a fixed point on the kinematic model instead of the true center of mass, we not only avoid having to provide masses and mass distributions to construct a physical model for the computation of the actual center of mass, but we also evade additional noise originating from every moving part of the body that has non-zero mass. We abstain from using quantities that are difficult to assess without high-quality sensors, such as torques, forces, and accelerations. It is not necessary to estimate the type and magnitude of a disturbance, e. g. the magnitude of an impulse. No matter what type of disturbance alters the trajectory of the fixed point—pushes, collisions, or the inertial effect of moving robot parts—the balance controller will react and suggest new step parameters.

It is easier and less noisy to track a fixed point on the body than to determine the true center of mass.

The tilted pose reconstruction can serve as a useful source of information for tasks other than walking and balancing. While the roll and pitch angle of the trunk is sufficient for balancing purposes, a global yaw estimation can be likewise integrated into the pose, when performing the rotation of the model about the support foot. Then, for example, a camera pose estimate can be gained from the model and used to seed visual registration methods. Finally, the sequence of footstep frames computed by the model during walking, and the CoM motion relative to the footstep frame, can be used as a dead-reckoning motion model to support localization.

8 ANALYTIC FOOTSTEP CONTROL

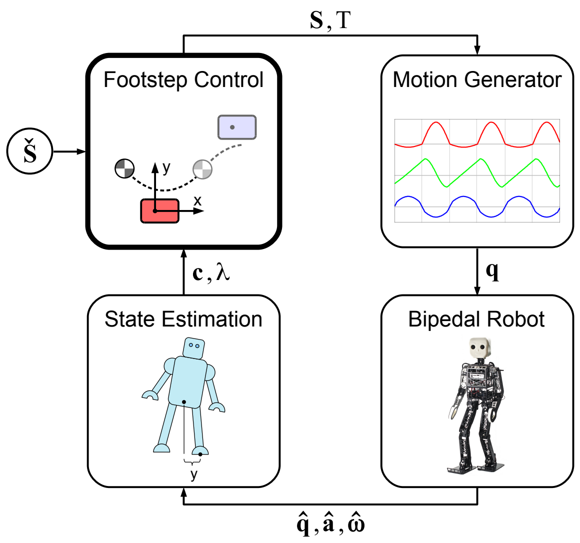


Figure 8.1: The Footstep Control module (top left) receives the Center of Mass state $\mathbf{c}$ and the support leg sign λ from the State Estimation (bottom left). A desired step size $\check{\mathbf{S}}$ from a higher control instance informs the Footstep Control about the target step location. Based on the Linear Inverted Pendulum Model, the Footstep Control computes the step size $\mathbf{S}$ and step time T for the Motion Generator.

Closed-loop balance control is imperative for a legged robot in order to remain upright for long periods of time in unconstrained environments, where disturbances of balance can frequently occur. We introduce an analytic footstep control method that augments the open-loop CPG with closed-loop disturbance rejection and reference tracking capabilities.

Due to the limited open-loop stability of bipeds, closed-loop control of balance is essential.

The Footstep Control module shown on the top middle in Figure 8.1 is an integral part of the control loop. The Footstep Control module executes the footstep control task to obey the desired step size $\check{\mathbf{S}} = (\check{S}_x, \check{S}_y, \check{S}_\psi)$ while preserving the balance of the robot. Unfortunately, these two goals can be mutually exclusive. The target step size can only be met if the balance of the robot is not compromised. If this is not the case, a footstep location must rather be chosen that prevents the robot from falling.

When a robot loses its balance, it must first avoid a fall before it can return to the commanded path.

The State Estimation component, shown in the bottom left in Figure 8.1, compresses the whole-body state of the robot into a point mass model, and provides the Center of Mass (CoM) state vector $\mathbf{c} = (c_x, \dot{c}_x, c_y, \dot{c}_y)$ along with the sign $\lambda \in \{-1, 1\}$ of the support

Only a low-dimensional point mass representation is used for balance control.

leg. The vector $\mathbf{c}$ contains positions and velocities of the CoM in the sagittal and lateral directions with respect to the support foot. The outputs of the Footstep Control are the coordinates $\mathbf{S} = (S_x, S_y, S_\psi)$ of the location where the swing foot should touch down with respect to the support foot, and the step time T, the remaining time until the next support exchange. The step parameters $(\mathbf{S}, T)$ are realized as a whole-body stepping motion by the Motion Generator shown in the top right in Figure 8.1.

Formally, the Footstep Control is a footstep control function

$$(\mathbf{S}, T) = \mathcal{F}(\check{\mathbf{S}}, \mathbf{c}, \lambda). \tag{8.1}$$

Zero Moment Point, step timing, and foot placement control strategies help the robot to maintain balance.

In its core, the Footstep Control generates an ideal reference trajectory for the Center of Mass. The reference trajectory is the limit cycle that the Center of Mass would follow under perfect conditions when executing stepping motions with the desired step size $\check{\mathbf{S}}$. The limit cycle is gained by walking the robot with the open-loop CPG and approximating the observed CoM trajectory with a parameterized Linear Inverted Pendulum Model. Robust and controllable walking performance is achieved by driving the Center of Mass towards the reference trajectory by means of analytically computed Zero Moment Point (ZMP), step timing, and foot placement control strategies.

An algorithmic representation of the analytic footstep control function $\mathcal{F}$ is presented in Algorithm 8.1. Three main computation steps can be identified within the FootstepControl($\check{\mathbf{S}}$, $\mathbf{c}$, λ) function. The first computation step is the application of a predictive filter that smooths the CoM state input and overcomes latency by means of prediction. The PredictiveFilter($\mathbf{c}$, λ) function overwrites the CoM state $\mathbf{c}$ and the support leg sign λ with a smoothed short-term prediction. The ReferenceTrajectory($\check{\mathbf{S}}$, λ) function computes a target state $\mathbf{s}$ that guides the CoM towards the limit cycle trajectory. The BalanceControl($\mathbf{s}$, $\mathbf{c}$, λ) function computes the ZMP $\mathbf{Z}$, the step size $\mathbf{S}$, and the step time T, which drive the CoM state $\mathbf{c}$ towards the target state $\mathbf{s}$ and make sure that the CoM does not diverge from the reference trajectory. Please note that only the step size $\mathbf{S}$ and the step

Algorithm 8.1 Footstep Control

```
Input: Desired step size $\check{\mathbf{S}}$                 ▷ Command input
Input: CoM state $\mathbf{c}$, support foot $\lambda$          ▷ From the State Estimation
Output: Step parameters $(\mathbf{S}, T)$                       ▷ Step size and timing
1: function FootstepControl($\check{\mathbf{S}}, \mathbf{c}, \lambda$)
2:     $(\mathbf{c}, \lambda) \leftarrow$ PredictiveFilter($\mathbf{c}, \lambda$)
3:     $\mathbf{s} \leftarrow$ ReferenceTrajectory($\check{\mathbf{S}}, \lambda$)
4:     $(\mathbf{Z}, \mathbf{S}, T) \leftarrow$ BalanceControl($\mathbf{s}, \mathbf{c}, \lambda$)
5:     return $(\mathbf{S}, T)$
6: end function
```

timing T are returned by the FootstepControl() function. The CPG does not need the ZMP for the generation of the stepping motion and the ZMP can remain hidden inside the Footstep Control component.

In the remainder of this chapter, we first introduce the Linear Inverted Pendulum Model in Section 8.1. This is the mathematical framework that we use to derive our step control formulae. We then walk through the computation steps in Algorithm 8.1 in sequential order. In Section 8.2, we introduce the predictive filter that we use to reduce noise and overcome control loop latency. We elaborate on the generation of the CoM reference trajectory in Section 8.3, and introduce the computation of step parameters for balanced reference tracking in Section 8.4. In Section 8.5, we present experimental results and finally, in Section 8.6, we summarize and discuss our results.

8.1 THE LINEAR INVERTED PENDULUM MODEL

8.1.1 *One-dimensional Model*

Figure 8.2a illustrates a one-dimensional Linear Inverted Pendulum Model (LIPM). The model was originally proposed by Kajita et al. [2001]. It is an approximation of an inverted pendulum and resembles a bipedal walker standing on one support leg, falling away from the pendulum base. The pendulum base, the ZMP, and the Center of Pressure (CoP), are equivalent concepts that label the pivot point of the pendulum on the ground. The quantity of our interest is the horizontal coordinate x of the CoM with respect to the pendulum base. The LIPM describes the motion of the CoM using the differential equation

The Linear Inverted Pendulum Model is a mathematical model that captures the principal dynamics of bipedal walking.

$$\ddot{x} = C^2 x \tag{8.2}$$

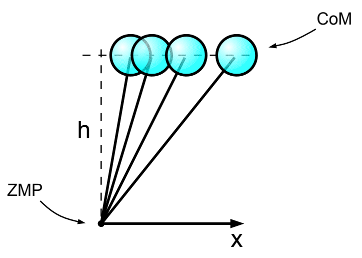
for some constant C. C is used in squared form to simplify the equations that are derived from equation (8.2). Typically, a value of $C = \sqrt{G/h}$ is used for C, where $G = 9.81\,\mathrm{m/s^2}$ is the gravitational constant and h is the assumed constant height of the center of mass. As in our approach it is not the LIPM that generates the walking motion, but a CPG, we can identify the value of C for an individual robot experimentally to fit the LIPM as closely as possible to the observed behavior.

The LIPM differential equation (8.2) has a closed form solution. For an initial state $(x_0, \dot{x}_0)$, the location and the velocity of the future state at time t are computed by

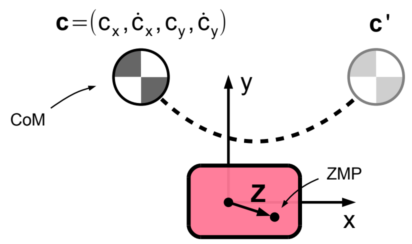
Closed form solutions for the prediction of future states are available.

$$x(t, x_0, \dot{x}_0) = x_0 \cosh(Ct) + \frac{\dot{x}_0}{C}\sinh(Ct), \tag{8.3}$$

$$\dot{x}(t, x_0, \dot{x}_0) = x_0 C \sinh(Ct) + \dot{x}_0 \cosh(Ct). \tag{8.4}$$



(a) 1D



(b) 2D

Figure 8.2: (a) The one-dimensional Linear Inverted Pendulum Model with pendulum height h and Center of Mass coordinate x. (b) Using the orthogonal superposition of two Linear Inverted Pendulum Models, the Center of Mass motion is approximated in a two-dimensional plane. Ideally, the Zero Moment Point is in the origin of the coordinate frame underneath the ankle joint. A small Zero Moment Point offset $\mathbf{Z}$ can be used for limited influence on the motion of the Center of Mass $\mathbf{c}$.

The time t when the CoM reaches a future location x, or velocity $\dot{x}$, is given by

$$t(x, x_0, \dot{x}_0) = \frac{1}{C} \ln \left(\frac{x}{c_1} \pm \sqrt{\frac{x^2}{c_1^2} - \frac{c_2}{c_1}} \right), \tag{8.5}$$

$$t(\dot{x}, x_0, \dot{x}_0) = \frac{1}{C} \ln \left(\frac{\dot{x}}{c_1 C} \pm \sqrt{\frac{\dot{x}^2}{c_1^2 C^2} + \frac{c_2}{c_1}} \right), \tag{8.6}$$

where

$$c_1 = x_0 + \frac{\dot{x}_0}{C}, \tag{8.7}$$

$$c_2 = x_0 - \frac{\dot{x}_0}{C}. \tag{8.8}$$

Unless the pendulum is disturbed by external forces, the orbital energy

$$E(x, \dot{x}) = \frac{1}{2} \left(\dot{x}^2 - C^2 x^2 \right) \tag{8.9}$$

is constant for an entire trajectory.

Equations (8.3) to (8.9) serve as a repertoire for the derivation of formulae that compute pendulum base locations to be used either as the Zero Moment Point, or as footstep coordinates, in order to steer the pendulum to a desired future state.

8.1.2 *Two-dimensional Model*

In accordance with the dimensional decomposition suggested in Chapter 5, we model a two-dimensional CoM motion using two uncoupled LIPM equations

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} C^2 & 0 \\ 0 & C^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{8.10}$$

A two-dimensional Linear Inverted Model simulates the motion of the Center of Mass under the influence of a constant Zero Moment Point offset.

The x dimension describes the sagittal motion and the y dimension describes the lateral motion. Additionally, we extend this passive model with ZMP control, as illustrated in Figure 8.2b. The origin of the coordinate frame is located underneath the ankle joint of the support foot. A small ZMP offset $\mathbf{Z} = (Z_x, Z_y)$ can relocate the pendulum base within the foot and influence the future motion of the CoM state $\mathbf{c} = (c_x, \dot{c}_x, c_y, \dot{c}_y)$. If we assume that the ZMP offset remains constant, we can use the one-dimensional LIPM state predictor equations (8.3) and (8.4), and incorporate the ZMP offset $\mathbf{Z}$ in a trivial manner into a two-dimensional LIPM predictor function. The resulting predictor function is presented in Algorithm 8.2. The PREDICT($\mathbf{c}$, $\mathbf{Z}$, t) function predicts the future CoM state $\mathbf{c'} = (c'_x, \dot{c}'_x, c'_y, \dot{c}'_y)$ at time t, given the current CoM state $\mathbf{c}$ at time $t = 0$, and the ZMP offset $\mathbf{Z}$. The prediction is computed efficiently in closed form and includes the effect of the ZMP control input. Typical use cases for the predictions are latency compensation and the computation of footstep locations based on estimated future states.

Algorithm 8.2 LIPM2D Predict

Input: CoM state $\mathbf{c}$, ZMP offset $\mathbf{Z}$, prediction time t
Output: Future CoM state $\mathbf{c'}$ at time t

```
1: function PREDICT($\mathbf{c}$, $\mathbf{Z}$, $t$)
2:     $c'_x \leftarrow x(t,\ c_x - Z_x,\ \dot{c}_x) + Z_x$        ▷ Eq. (8.3)
3:     $\dot{c}'_x \leftarrow \dot{x}(t,\ c_x - Z_x,\ \dot{c}_x)$        ▷ Eq. (8.4)
4:     $c'_y \leftarrow x(t,\ y_x - Z_y,\ \dot{c}_y) + Z_y$        ▷ Eq. (8.3)
5:     $\dot{c}'_y \leftarrow \dot{x}(t,\ c_y - Z_y,\ \dot{c}_x)$        ▷ Eq. (8.4)
6:     $\mathbf{c}' \leftarrow (c'_x, \dot{c}'_x, c'_y, \dot{c}'_x)$
7:     return $\mathbf{c}'$
8: end function
```

8.2 PREDICTIVE FILTER

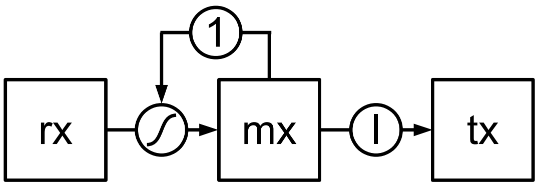


Figure 8.3: The Predictive Filter. This filter removes noise by blending between a measured Center of Mass state $\mathbf{rx}$ and an expected state $\mathbf{mx}$. The filter also predicts a short-term future state $\mathbf{tx}$ to overcome the latency.

In a real hardware environment, issues originating from sensor noise and latency cannot be neglected.

In a real hardware environment, sensor noise and latency in the control loop can have a significant effect on the performance of control algorithms. A Predictive Filter is an integral part of the Footstep Control algorithm 8.1 to remove noise from the CoM state estimate and to compensate the latency by making a short-term prediction with the LIPM. The filter is illustrated in Figure 8.3. Its three building blocks are denoted $\mathbf{rx}$, $\mathbf{mx}$, and $\mathbf{tx}$. The $\mathbf{rx}$ block encapsulates the CoM state computed from the raw sensor input by the State Estimation. The second building block named $\mathbf{mx}$ is a model state. In every iteration of the main control loop, the $\mathbf{mx}$ state is forwarded by the time period of one iteration of the control loop using the LIPM. The forwarded state is then linearly interpolated with the $\mathbf{rx}$ state using a blending factor $b \in [0, 1]$ and written back into the $\mathbf{mx}$ state. In this manner, the $\mathbf{rx}$-$\mathbf{mx}$ loop forms a noise filter that blends between an expected state according to the LIPM, and a raw input state estimated from the sensor input. The $\mathbf{tx}$ block contains the $\mathbf{mx}$ state forwarded in time by the latency l, once again using the LIPM equations. It is the $\mathbf{tx}$ CoM state that is presented to the BALANCECONTROL computation step of the Footstep Control algorithm 8.1. Effectively, the footstep controller does not compute the step parameters for the state that was last measured, but instead for the state the robot is estimated to be in by the time the motors execute the commands.

Noise and latency can both be compensated with the help of predictions made by the Linear Inverted Pendulum Model.

For a detailed description of the Predictive Filter algorithm, we define the $\mathbf{rx}$ state to be a vector $\mathbf{rx} = (\mathbf{c}_{rx}, \lambda_{rx}, t_{rx}, T_{rx})$ that contains the CoM state $\mathbf{c}_{rx}$ and the support leg sign λ_{rx}, as they were computed by the State Estimation module. t_{rx} is the time that has passed since the last detected support exchange, and T_{rx} is the estimated time until the next support exchange. We similarly define the $\mathbf{mx}$ state $\mathbf{mx} = (\mathbf{c}_{mx}, \lambda_{mx}, t_{mx}, T_{mx})$, and the $\mathbf{tx}$ state $\mathbf{tx} = (\mathbf{c}_{tx}, \lambda_{tx}, t_{tx}, T_{tx})$.

The computation steps of the Predictive Filter are shown in Algorithm 8.3. Line 3 computes the linear blending between the $\mathbf{rx}$ CoM state $\mathbf{c}_{rx}$ and the $\mathbf{mx}$ CoM state $\mathbf{c}_{mx}$, which has been forwarded in line 2 by the system iteration time $\rho = 0.01$ s using the ForwardMX algorithm 8.4. The ForwardMX algorithm encapsulates the two-dimensional LIPM model with a ZMP control input, which we introduced

Algorithm 8.3 Predictive Filter

```
Input: CoM state $c_{rx}$, support foot $\lambda_{rx}$          ▷ From the State Estimation
Output: CoM state $c_{tx}$, support foot $\lambda_{tx}$          ▷ Smoothed and predicted
 1: function PREDICTIVEFILTER($c, \lambda$)
 2:     $mx \leftarrow$ FORWARD($mx, Z, \rho$)
 3:     $c_{mx} \leftarrow b * c_{rx} + (1 - b) * c_{mx}$          ▷ Noise filter
 4:     $(T, Z, S) \leftarrow$ BALANCECONTROL($c_{mx}, \lambda_{mx}$)
 5:     $T_{mx} \leftarrow T$
 6:     if ($T < l$) then
 7:         $tx \leftarrow$ FORWARD($mx, Z, T$)          ▷ Alg. 8.4
 8:         $tx \leftarrow$ STEP($tx, S$)          ▷ Alg. 8.5
 9:         $tx \leftarrow$ FORWARD($tx, Z, l - T$)          ▷ Alg. 8.4
10:     else
11:         $tx \leftarrow$ FORWARD($mx, l$)          ▷ Alg. 8.4
12:     end if
13:     if ($T < \rho$) then
14:         $mx \leftarrow$ STEP($mx, \lambda$)          ▷ Alg. 8.5
15:     end if
16:     $b \leftarrow \delta_{\lambda_{rx}, \lambda_{mx}} * SNS * DEX$          ▷ Eq. (8.13)
17:     return $(c_{tx}, \lambda_{tx})$
18: end function
```

Algorithm 8.4 ForwardMX

```
Input: Model state $mx$, ZMP offset $Z$, time $t$
Output: Model state $mx'$          ▷ Forwarded in time
1: function FORWARD($mx, Z, t$)
2:     $c'_{mx} \leftarrow$ PREDICT($c_{mx}, Z, t$)          ▷ LIPM2D predict 8.2
3:     $\lambda'_{mx} \leftarrow \lambda_{mx}$
4:     $t'_{mx} \leftarrow t_{mx} + t$
5:     $T'_{mx} \leftarrow T_{mx} - t$
6:     return $mx'$
7: end function
```

Algorithm 8.5 Step

```
Input: Model state $mx = (c, \lambda, t, T)$          ▷ Pre step state
Input: Step size $S = (S_x, S_y, S_\psi)$          ▷ Planned step size
Output: Model state $mx' = (c', \lambda', t', T')$          ▷ Post step state
1: function STEP($mx$, $S$)
2:     $c'_x \leftarrow S_x/2$          ▷ Symmetric step assumption
3:     $\dot{c}'_x \leftarrow \dot{c}_x$          ▷ Constant velocity assumption
4:     $c'_y \leftarrow S_y/2$
5:     $\dot{c}'_y \leftarrow \dot{c}_y$
6:     $\lambda' \leftarrow -\lambda$          ▷ Fip the leg sign
7:     $t' \leftarrow 0$          ▷ Reset time since step
8:     return $mx'$
9: end function
```

in Section 8.1.2. The ZMP used for the forwarding of the $\mathbf{mx}$ state in line 2 bears only marginal significance, since the prediction is made only over a very short time horizon. A good value to use is the ZMP $\mathbf{Z}$ that was computed in the previous iteration in line 4. The blending factor b is computed at a later time in the algorithm. We will turn our attention to it after having discussed the latency compensation.

Due to the high latency, the generation of the motion for the next step has to begin before the support exchange is detected.

We determined a latency of $l = 65\,\text{ms}$ on the robots we used for experimentation. This is quite significant considering that an undisturbed step takes approximately $420\,\text{ms}$. One portion of the latency is the time needed for the data communication between the main processing unit and the hardware. Another significant portion can be attributed to our unique setting of compliant actuation. The unfortunate implication of the high latency is that the support exchange has to be anticipated ahead of time, and the first portion of the motion signal for the next step has to be commanded, well before the actual support exchange is detected by the sensors. Consequently, when computing the latency-compensated $\mathbf{tx}$ state, a step may have to be included in the prediction. In line 4 of Algorithm 8.3, a set of step parameters $(T, \mathbf{Z}, \mathbf{S})$ is computed with the balance controller based on the $\mathbf{mx}$ state, which has not been forwarded by the latency yet. The computation of the step parameters is described in Section 8.4 and is here simply referred to as BALANCECONTROL($\mathbf{c}$, λ). The set of step parameters gained from the $\mathbf{mx}$ model is used to compute the $\mathbf{tx}$ state in lines 6 to 12. If the latency l exceeds the estimated remaining time T until the step, an anticipated support exchange must be included in the prediction of the $\mathbf{tx}$ state. In line 7, the $\mathbf{mx}$ model is forwarded with the ForwardMX algorithm 8.4 by the step time T to the moment of the anticipated support exchange. In line 8, the modeled support exchange is performed and a post step state is estimated with the STEP($\mathbf{mx}$, $\mathbf{S}$) function given in Algorithm 8.5. In line 9, the post step state is forwarded by the remaining time to the latency horizon. If the step time T is greater than the latency l, the $\mathbf{tx}$ model does not need to step and can be directly forwarded by the latency l in line 11.

Using the coordinate frame of the swing foot shortly before the step is a good way to predict the state of the Center of Mass after the step.

In the STEP($\mathbf{mx}$, $\mathbf{S}$) function in Algorithm 8.5, the support exchange is modeled as an instantaneous change of the CoM location without a double support phase. The CoM velocity is assumed to be unaffected by the step. A safe way to estimate the coordinates of the CoM after the step is to rely on the symmetrical step assumption and to use half of the step size for the post-step CoM coordinates, as indicated in lines 2 and 4. This method can be used when the blending factor b is set to zero and the $\mathbf{mx}$-$\mathbf{tx}$ states drive the walking motion in an open-loop mode. A more precise method is to use the coordinates of the CoM in the ground projected frame of the swing foot as a post-step estimate. When the estimated step time T is smaller than the latency l, the swing foot is typically close to its landing position and the coordinates of the CoM in the ground projected swing foot frame are a good estimate of the post-step state. However, this method closes the feedback loop and must be used with care.

When the estimated step time decrements to a value $T < \rho$, the $\mathbf{mx}$ model is stepped in line 14 of the Predictive Filter algorithm 8.3, whether the actual support exchange has been detected or not. This means that at times near the support exchange, the $\mathbf{rx}$ state and the $\mathbf{mx}$ state may have different support leg signs and cannot reasonably be blended.

The blending factor b is a powerful parameter that determines the extent to which the sensor input can influence the internal state of the system. If the blending factor is set to $b = 0$, the sensor input is ignored entirely and the gait generation process runs in open-loop mode. The LIPM simulation of the $\mathbf{mx}$ state reproduces the reference trajectory. The robot will still attempt to step to the commanded step location $\check{\mathbf{S}}$ and is therefore controllable, but it will not adapt to disturbances. In the other extreme, when the blending factor is set to $b = 1$, the $\mathbf{rx}$ state overwrites the $\mathbf{mx}$ state in every cycle and the loop is completely closed. No reduction of noise takes place. This is not recommended, because noise can destabilize the system and stimulate it to produce even more noise. For values $0 < b < 1$, the blending works similar to a Kalman filter. The filter output is a mixture of an expected state computed by the LIPM simulation, and a state that was estimated based on the sensor input.

Blending between the measured and a model predicted state of the system is an efficient way to remove noise.

We take active control of the blending factor during walking, and dynamically adjust its value according to the following rules:

Active control of the blending factor enables domain-specific filtering.

- In the case where the $\mathbf{rx}$ and the $\mathbf{mx}$ states have different support leg signs, they cannot be blended, and the blending factor is set to $b = 0$.

- We inhibit the adaptation and smoothly decrease b with a Gaussian step noise suppression function when the estimated step time T approaches zero, and allow it to rise again after the support exchange. We compute the step noise suppression factor

$$\mathrm{SNS} = 1 - \exp\left(-\frac{\min\{t_{mx}, T_{mx}, T_{tx}\}^2}{2\epsilon^2}\right), \qquad (8.11)$$

  where $\epsilon = 0.07$ is a tuning parameter. With this mechanism in place, short periods of open-loop motion are used to discard excessive sensor noise near the support exchange, and to bridge a short period of physical double support motion that can deviate from the LIPM model.

- Finally, we decrease the value of b when the Euclidean distance between the $\mathbf{c}_{rx}$ and the $\mathbf{c}_{mx}$ CoM states is small. This indicates that the CoM trajectory is developing as expected and there is

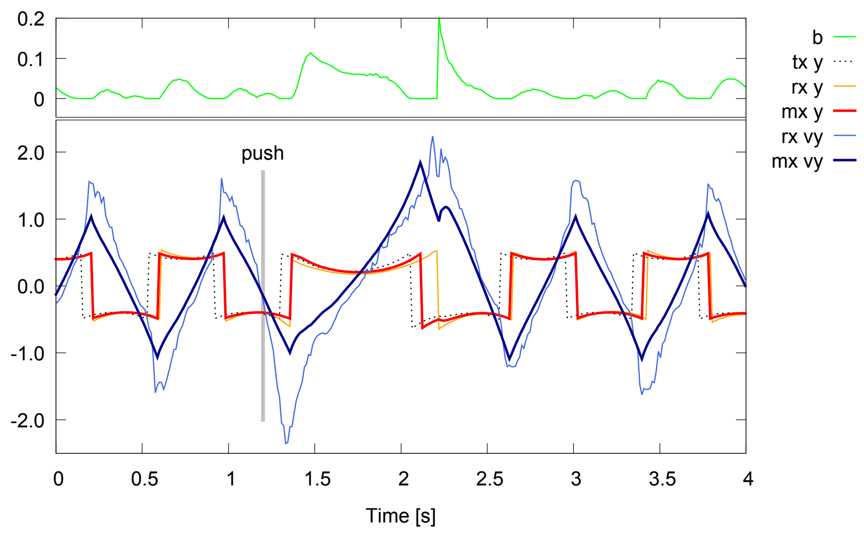


Figure 8.4: Demonstration of the effects of the predictive filter in an experiment with Dynaped. We observe the commanded (tx) lateral position of the Center of Mass, the measured (rx) lateral position and velocity of the Center of Mass, the filtered (mx) lateral position and velocity of the Center of Mass, and the blending factor b while the robot is walking in place. The vertical gray line indicates a moment when the robot was pushed from the side. The blending factor b rises and the filter adapts to the new trajectory. The filter successfully discards the undesired peaks of the measured Center of Mass velocity around the support exchanges.

no need for adaptation. The deviation from the expected state is expressed as

$$\mathrm{DEX} = k\|\mathbf{c}_{rx} - \mathbf{c}_{mx}\|, \tag{8.12}$$

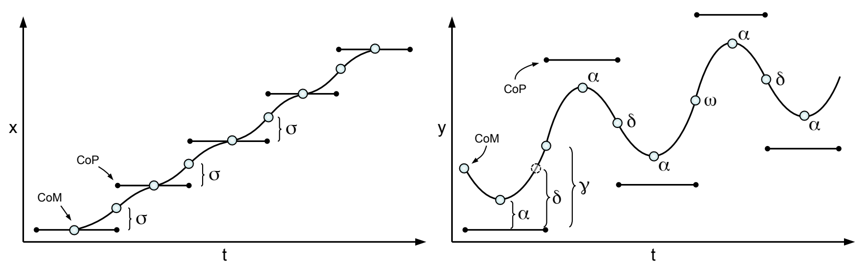
where $k = 0.5$ is a gain. This way, the gait controller has a tendency towards following an open-loop trajectory when the state of the system develops as expected, and avoids the possibly destabilizing effects of sensor noise.

The blending factor b is then computed with the function

$$b = \delta_{\lambda_{rx},\lambda_{mx}} \cdot \mathrm{SNS} \cdot \mathrm{DEX}, \tag{8.13}$$

where $\delta_{\lambda_{rx},\lambda_{mx}}$ is the Kroneker delta applied to λ_{rx} and λ_{mx}. The computation of the blending factor is embedded into the Predictive Filter algorithm 8.3 in line 16.

Figure 8.4 shows the commanded (tx) lateral position of the Center of Mass, the measured (rx) lateral CoM position and velocity, and the filtered (mx) lateral CoM position and velocity recorded during an experiment with Dynaped. While the robot was walking on the spot, it was pushed from the side in the moment indicated by the gray vertical bar. The smoothing effect of the predictive filter can best be seen by comparing the **rx** and **mx** velocity data. The filter discards the



(a) Sagittal (b) Lateral

Figure 8.5: The two-dimensional Center of Mass reference trajectory is composed of (a) a sagittal motion and (b) a lateral motion. Four configuration parameters define the maximum sagittal Center of Mass displacement σ, the lateral apex distance α, and the minimal and maximal support exchange locations, δ and γ, in the lateral direction. The support exchange is modeled as an instantaneous relocation of the pendulum base such that the Center of Mass is in the center between the pivot points before and after the support exchange.

high velocity peaks at the support exchange that differ strongly from the pendulum model and lead to bad predictions. The blending factor shows its highest peaks right after the push and after the capture step. The model adapts nicely to the new pendulum trajectory caused by the push, yet eliminates the jittery noise after the 2.0 second mark. At the first step after the push, the rx and mx signals are slightly out of synchronization. The mx model steps earlier than the robot, but synchronization is quickly restored.

In the following section, we discuss the next computation step of the Footstep Control algorithm 8.1, which is the generation of a reference trajectory for the Center of Mass.

8.3 REFERENCE TRAJECTORY GENERATION

The gait generation cycle leans on the computation of a nominal trajectory, which is a limit cycle that describes an ideal motion of the CoM. The shape of the nominal trajectory is determined by the desired step size $\check{S}$ and a set of constant parameters. It does not depend on the current state of the CoM.

The reference trajectory is a limit cycle that describes the ideal motion of the Center of Mass.

The nominal trajectory is composed by orthogonal superposition of two uncoupled, one-dimensional LIPMs. Figure 8.5 shows trajectory schematics for the sagittal and lateral dimensions. In the sagittal direction, the point mass crosses the pendulum base in every step cycle. In the lateral direction, however, the point mass oscillates between two supports and never crosses the pendulum base. Under the ideal conditions of the limit cycle trajectory, the pendulum base is assumed to stay stationary during a step, and to instantly relocate in the moment of the support exchange in a way that the last position of the CoM at the end of the step is in the center between the pendulum bases before and after the relocation. A double support phase is not included in our nominal trajectory model.

The reference trajectory is characterized by parameters derived from a rough concept of the walking motion.

Based on the underlying principles of bipedal walking we introduced in Chapter 5, we identify four parameters that characterize the walking motion. The parameters are illustrated in Figure 8.5. The lateral distance between the pivot point and the lateral apex of the point mass trajectory is denoted α. It is evident that the lateral component of the CoM velocity is zero at this point. As long as the apex distance is greater than zero, the point mass will return from the foot and reach the lateral support exchange location in a range bounded by δ and γ. When walking in place, we assume the support exchange occurs at the distance δ. When walking with a non-zero lateral velocity, the walker first takes a long step with the leading leg and then a shorter trailing step. The support exchange at the end of the leading step occurs at a distance between δ and an upper bound γ, depending on the desired lateral step size. The trailing step is assumed to result in a support exchange at δ, independent of the size of the leading step. For the sagittal direction, one parameter is sufficient to describe the Center of Mass motion. σ defines an absolute bound for displacement of the CoM with respect to the foot. The CoM displacement is negative when the robot walks backwards.

The reference trajectory is fitted to the limit cycles observed during open-loop walking.

We estimate the values of the reference trajectory parameters α, δ, γ, and σ using data collected from a robot walking with the open-loop CPG. We averaging the lateral apex coordinates, the lateral coordinates of the support exchange points when walking in place, and when walking with full lateral velocity, and the sagittal CoM displacement when walking forward with full velocity, respectively. The CoM displacement limit σ can be increased once a reliable balance controller has been achieved.

The reference trajectory is expressed with a single nominal state at the end of the step.

For the mathematical formulation of the reference trajectory, we exploit the assumptions that the motion of the CoM follows the laws of the LIPM and that the Center of Pressure remains stationary during the step in the ideal case. With these conditions in effect, a single state $\mathbf{s} = (s_x, \dot{s}_x, s_y, \dot{s}_y)$ is sufficient to represent the limit cycle. We choose the state $\mathbf{s}$ to be the end-of-step CoM state, that is, the CoM state in the moment of the support exchange. In the following, we introduce the formulas for the computation of the limit cycle representative $\mathbf{s}$.

Given the configuration parameters α, δ, γ, and σ, the pendulum constant C (8.2), and the desired step size $\check{\mathbf{S}} = (\check{S}_x, \check{S}_y, \check{S}_\psi)$, we first compute the desired lateral support exchange location ξ_y. We differentiate between the leading step case, where the lateral support exchange location ξ_y is bounded by δ and γ, and the trailing step case, where the lateral support exchange always occurs at distance δ. The lateral support exchange location is given by

$$\xi_y = \begin{cases} \min(\max(\frac{|\check{S}_y|}{2}, \delta), \gamma), & \text{if } \lambda = \operatorname{sgn}(\check{S}_y) \\ \delta, & \text{otherwise,} \end{cases} \tag{8.14}$$

where $\lambda \in \{-1, 1\}$ denotes the sign of the support leg. Due to the symmetrical step assumption, the lateral support exchange location is trivially half of the desired leading step size, bounded to the permitted range $[\delta, \gamma]$. Please note that ξ_y is always positive due to the fact that the configuration parameters have positive values. Later on, the support leg sign λ will be used to obtain the correct sign for ξ_y.

The computation of the bounded sagittal support exchange location

$$\xi_x = \min(\max(\frac{\check{S}_x}{2}, -\sigma), \sigma) \tag{8.15}$$

is also trivial due to the symmetric step assumption.

Both coordinates of the support exchange location have to be reached at the same time, but it is the relatively constrained lateral motion that determines the time period of the steps. We introduce the half step time variable τ, which is the time it takes for the Center of Mass to travel from the lateral apex α to the lateral support exchange coordinate ξ_y. The nominal half step time is computed as

$$\tau = \frac{1}{C} \ln\left(\frac{\xi_y}{\alpha} + \sqrt{\frac{{\xi_y}^2}{\alpha^2} - 1} \right). \tag{8.16}$$

To determine τ, we use $(\alpha, 0)$ as the initial state and solve the LIPM time-of-location equation $\tau = t(\xi_y, \alpha, 0)$ (8.5) to compute the time it takes to reach the lateral support exchange location ξ_y.

We can now completely express the nominal support exchange state $\mathbf{s}$ as

$$\mathbf{s} = \begin{bmatrix} s_x \\ \dot{s}_x \\ s_y \\ \dot{s}_y \end{bmatrix} = \begin{bmatrix} \xi_x \\ \xi_x \, C \coth(C\tau) \\ \lambda \xi_y \\ \lambda C \sqrt{s_y^2 - \alpha^2} \end{bmatrix}. \tag{8.17}$$

The nominal state $\mathbf{s}$ is expressed in coordinates relative to the current support foot. The coordinates (s_x, s_y) of the nominal state are identical with the coordinates of the support exchange location (ξ_x, ξ_y), apart from the leg sign λ that disambiguates the lateral support exchange coordinate. To compute the sagittal support exchange velocity $\dot{s}_x$, we solve for $\dot{x}_0$ the LIPM location prediction equation $x(\tau, 0, \dot{x}_0) = \xi_x$ (8.3) with the initial state $(0, \dot{x}_0)$, and the known output location

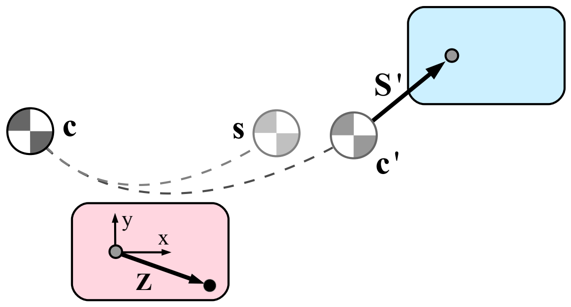


Figure 8.6: The balance controller computes a Zero Moment Point offset $\mathbf{Z}$ that steers the Center of Mass $\mathbf{c}$ towards the nominal support exchange state $\mathbf{s}$. The Zero Moment Point is not always effective in reaching the target state, but the achievable end-of-step state $\mathbf{c}'$ can be predicted. The location of the next footstep $\mathbf{S}'$ is computed with respect to the achievable state $\mathbf{c}'$ and then converted to the foot-to-foot step size $\mathbf{S}$.

ξ_x. This yields the CoM velocity $\dot{x}_0$ at the sagittal apex. Then, we use the LIPM velocity prediction equation $\dot{x}(\tau, 0, \dot{x}_0) = \dot{s}_x$ (8.4) to compute $\dot{s}_x$ at the support exchange location. The computation of the lateral support exchange velocity $\dot{s}_y$ uses the law of conservation of energy (8.9) in conjunction with the orbital energy of the apex state $(\alpha, 0)$. We solve the equation $E(\xi_y, \dot{s}_y) = E(\alpha, 0)$ for $\dot{s}_y$.

The computation of the nominal state $\mathbf{s}$ using Equation (8.17) is equivalent to the ReferenceTrajectory($\check{\mathbf{S}}, \lambda$) computation step in line 3 of the Footstep Control algorithm 8.1.

8.4 BALANCE CONTROL

In any given state, the balance controller attempts to reach the nominal state by means of Zero Moment Point control, and computes the footstep location based on a predicted achievable state by the end of the step.

The next computation step of the Footstep Control algorithm 8.1 is the BalanceControl($\mathbf{s}, \mathbf{c}, \lambda$) function in line 4. Given the nominal target state $\mathbf{s}$, the filtered and latency-compensated CoM state $\mathbf{c}$, and the sign λ of the support leg, the balance control function computes the ZMP offset $\mathbf{Z}$, the step size $\mathbf{S}$, and the step timing T parameters that make the robot track the commanded step size $\check{\mathbf{S}}$ while maintaining balance. Inside the balance controller, the nominal state $\mathbf{s}$ represents the limit cycle trajectory that would reproduce the commanded step size $\check{\mathbf{S}}$. Aiming to make the CoM state $\mathbf{c}$ reach the nominal state $\mathbf{s}$ by the end of the step is equivalent to tracking the commanded step size. Disturbances can force the walker away from the nominal trajectory and necessitate a choice of footstep location that differs from the commanded one. The balance controller is prepared to adapt the step timing and the step size in order to absorb a disturbance, and to return to the reference trajectory in a suitably short time.

The concept of the balance controller is illustrated in Figure 8.6. The balance controller computes a ZMP offset $\mathbf{Z}$ that steers the CoM towards the target location $\mathbf{s}$. The ZMP offset $\mathbf{Z}$ is expressed relative to the ankle joint of the support leg. Since the ZMP is physically bounded to remain inside the support polygon, the effect of the ZMP is limited and the target state is not guaranteed to be reached. The time T for the support exchange is chosen to be the time when the CoM reaches the lateral coordinate of the target state. Based on the support exchange time T and the bounded ZMP offset $\mathbf{Z}$, the balance controller predicts the achievable end-of-step state $\mathbf{c}'$ and uses it to compute the step coordinates $\mathbf{S}'$ expressed with respect to the predicted state $\mathbf{c}'$. Finally, the step coordinates $\mathbf{S}'$ are converted to a foot-to-foot step size $\mathbf{S}$. For the computation of the aforementioned step parameters, closed-form formulae are derived from the LIPM.

The dimensional decomposition introduced in Chapter 5 simplifies the task of computing the step parameters by splitting the CoM motion into uncoupled, one-dimensional entities. It allows us to employ different strategies for the computation of sagittal and lateral parameters. The following sections outline, in order, how the various step parameters are computed. The order is important as later parameters depend on the values of the former ones.

8.4.1 *Lateral Zero Moment Point Offset*

The ZMP offset exerts a limited amount of control over the CoM trajectory during the step by relocating the pivot point of the inverted pendulum from the ankle joint to a location within the support foot. The balance controller makes active use of this relocation and attempts to steer the CoM towards the well-defined target state $\mathbf{s}$. However, even in a one-dimensional setting, the transfer of a pendulum state $(c_y, \dot{c}_y)$ to a target state $(s_y, \dot{s}_y)$ in a desired time $\check{T}$ is not a simple control task. Since we are targeting compliant and imprecise hardware that is not fitted with the necessary sensors to measure the actual location of the ZMP, a complex control strategy that involves a smooth motion of the ZMP throughout the step cannot be enforced. We opt for a simple control strategy based on the assumption that if the CoM continues to travel along an undisturbed Linear Inverted Pendulum trajectory, the ZMP offset stays constant for the remainder of the step. It is not possible to satisfy all three target conditions—the location s_y, the velocity $\dot{s}_y$, and the time $\check{T}$—with this control paradigm. Only two out of the three conditions can be met. However, we gain an approximate solution in closed-form and preserve closed-form predictability of future CoM states.

The Zero Moment Point control is based on the assumption that the Zero Moment Point stays constant during a step.

Given the current CoM state $\mathbf{c} = (c_x, \dot{c}_x, c_y, \dot{c}_y)$ and the nominal support exchange location $\mathbf{s} = (s_x, \dot{s}_x, s_y, \dot{s}_y)$, we compute the lateral ZMP offset

$$Z_y = \frac{c_y \cosh(C\check{T}) + \frac{\dot{c}_y}{C}\sinh(C\check{T}) - s_y}{\cosh(C\check{T}) - 1} \tag{8.18}$$

that accelerates the CoM so that it reaches the lateral support exchange location s_y at the nominal step time $\check{T}$. To derive the formula above, we solved for the unknown ZMP offset Z_y the LIPM location predictor equation (8.3)

$$x(\check{T}, c_y - Z_y, \dot{c}_y) = s_y - Z_y. \tag{8.19}$$

$\check{T}$ is the nominal remaining step time, obtained by setting it to $\check{T} = 2\tau$ whenever a support exchange occurs, and decrementing it by $\rho = 0.01$s in every iteration, the time period of a 100 Hz control loop. τ (see Eq. (8.16)) is the half step time of the reference trajectory. When $\check{T}$ approaches 0, the result of the ZMP computation is increasingly unstable. $\check{T}$ can also have a negative value if the nominal step time is exceeded. The ZMP control formula (8.18) will still compute a sensible value, but Z_y has to be bounded to a reasonable range $[Z_y^{min}, Z_y^{max}]$, for example the width of the foot.

With this control approach, the lateral ZMP helps to maintain the nominal step frequency of the limit cycle by reaching the target location at the right time. However, due to the bounding of the ZMP offset, a constant step frequency cannot be guaranteed. The target lateral velocity $\dot{s}_y$ at the support exchange location is neglected, but a possible error in the lateral velocity at the support exchange location is corrected by the choice of the lateral step size.

A similar ZMP control approach was suggested by Englsberger et al. [2011]. In the same manner, a ZMP offset is selected that remains constant for the remainder of the step, assuming the point mass continues to travel along a perfect Linear Inverted Pendulum trajectory. However, instead of the location and the time, the capture point of the center of mass is made to match a nominal capture point at a given time $\check{T}$. The capture point $p_y = c_y + \frac{\dot{c}_y}{C}$ is a function of the CoM location c_y and velocity $\dot{c}_y$. Thus, this approach constrains the CoM to have one of an infinite set of position and velocity combinations by the end of the step, lying on the curve of a constant nominal capture point. When simulating the behavior of the capture point-based ZMP controller, we found that it often led to oscillatory behavior where the ZMP never settled. With our formula, the oscillatory behavior has not been observed.

8.4.2 *Step Time*

The best time to step is when the robot returns to the center of the lateral oscillation.

The next step parameter to compute is the step time T. Motivated by the observed sensitivity of the lateral oscillation, as demonstrated in video [3], we assume the lateral oscillation to be the main determinant of the step time. In most cases, the best time for the support exchange is when the CoM reaches the nominal lateral support exchange location s_y. In this position, the robot can be expected to be upright and to have sufficient lateral momentum to transfer its weight to the other leg. The ideal case is illustrated in Figure 8.7a. The CoM travels towards the support leg, passes through the apex, and returns. Even-

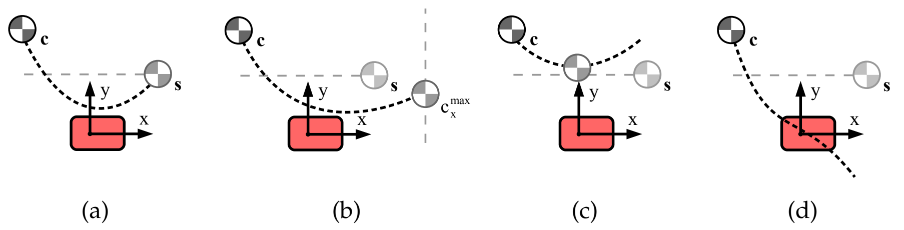


Figure 8.7: The balance controller estimates the remaining time of the step as the time when (a) the Center of Mass $\mathbf{c}$ reaches the lateral coordinate of the target location $\mathbf{s}$. Special cases, such as (b) reaching the sagittal limit c_x^{max} first, (c) never reaching the lateral coordinate, or (d) crossing the support foot, are handled explicitly.

tually it reaches the nominal support exchange location s_y. Using the LIPM time-of-location equation (8.5), the time to reach s_y is given by

$$t(s_y) = \frac{1}{C} \ln \left(\frac{s_y - Z_y}{c_1} + \sqrt{\frac{(s_y - Z_y)^2}{c_1^2} - \frac{c_2}{c_1}} \right), \tag{8.20}$$

where

$$c_1 = c_y - Z_y + \frac{\dot{c}_y}{C}, \tag{8.21}$$

$$c_2 = c_y - Z_y - \frac{\dot{c}_y}{C}. \tag{8.22}$$

To account for the previously computed lateral ZMP offset, we have subtracted it from the current CoM location c_y and the target location s_y.

There are, however, special cases, where the step time must be determined in a different way. Figure 8.7b shows the case, where a sagittal limit c_x^{max} is reached first. A push in the lateral direction can double or triple the time needed to return to the support exchange location. If during this time the CoM continues to move in the sagittal direction, a limit can be reached, beyond which an increase of the stride length would also compromise balance. The time to reach c_x^{max} is given by

$$t(c_x^{max}) = \frac{1}{C} \ln \left(\frac{c_x^{max}}{c_x + \frac{\dot{c}_x}{C}} + \sqrt{\frac{(c_x^{max})^2}{(c_x + \frac{\dot{c}_x}{C})^2} - \frac{c_x - \frac{\dot{c}_x}{C}}{c_x + \frac{\dot{c}_x}{C}}} \right). \tag{8.23}$$

Figure 8.7c depicts the case where the support exchange location s_y is never reached. A strong disturbance cause the CoM to never cross the support exchange coordinate. Situations where the support exchange location has been crossed in the past without a step having occurred also belong to this category. Case (c) can be detected when $t(s_y)$ (8.20) does not compute a positive value. In that case, if a positive time

$$t(0) = \frac{1}{C} \ln \left(\sqrt{\frac{c_y - Z_y - \frac{\dot{c}_y}{C}}{c_y - Z_y + \frac{\dot{c}_y}{C}}} \right) \tag{8.24}$$

can be determined using the LIPM time-of-velocity equation (8.6) with a target velocity of zero, then an irregular lateral apex is still to be encountered in the future. The irregular apex is the closest point to the lateral support exchange location, and the time to reach this apex can be used as a sensible step time. Otherwise, we set the step time to zero and the balance controller recommends an immediate step, which drives the step motion generator at its maximum permitted frequency towards the next support transition.

Finally, Figure 8.7d shows the critical case where the CoM is estimated to tip over the support foot, indicated by a positive lateral orbital energy $E(c_y, \dot{c}_y)$ (8.9). In this case, we use a large constant step time of $T = 2$ seconds to slow the stepping motion down and hope that the CoM will return after all. If the robot tips over, a recovery step cannot reasonably be taken, but at least the imminent fall can be predicted ahead of time. In this situation, the trunk, arms, and swing leg could be used to increase the odds of a return.

All cases considered, the step time parameter T is computed as

$$T = \begin{cases} t(c_x^{max}), & \text{if } t(c_x^{max}) < t(s_y), \\ t(s_y), & \text{if } t(s_y) > 0 \wedge t(s_y) < \infty, \\ t(0), & \text{if } t(0) > 0 \wedge t(s_y) = \infty, \\ 2, & \text{if } E(c_y, \dot{c}_y) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{8.25}$$

The step parameters computed in the following sections depend on the step time T.

8.4.3 *Sagittal Zero Moment Point Offset*

The same Zero Moment Point control concept is applied in the sagittal and lateral directions.

For the computation of the sagittal ZMP offset, we use the same control approach as in the lateral direction. We calculate a ZMP such that if the CoM continues to move along a Linear Inverted Pendulum trajectory, the ZMP offset stays constant and two out of the three target constraints—the position s_x, the velocity $\dot{s}_x$, and the step time T—are satisfied by the end of the step. In the sagittal direction, we also choose the position and time as the two conditions to satisfy, and compute the sagittal ZMP offset as

$$Z_x = \frac{c_x \cosh(CT) + \frac{\dot{c}_x}{C} \sinh(CT) - s_x}{\cosh(CT) - 1}. \tag{8.26}$$

To derive this formula we solve for the unknown ZMP offset Z_x the LIPM location predictor equation (8.3)

$$x(T, c_x - Z_x, \dot{c}_x) = s_x - Z_x. \tag{8.27}$$

Please note that in the sagittal direction, we aim for the CoM to arrive at the sagittal support exchange location at the predicted step time T, unlike in the lateral direction, where we aimed for the nominal step time $\check{T}$. This is how the sagittal motion accommodates variations in step timing, while the lateral motion attempts to maintain a nominal frequency.

Due to physical limitations, the sagittal ZMP offset must be bounded to a reasonable range $[Z_x^{min}, Z_x^{max}]$. The offsets to the forefoot and the heel are good initial values for the upper and lower ZMP bounds.

8.4.4 *Footstep Location*

Foot placement is the most powerful strategy to maintain balance.

The choice of the next footstep location is the most powerful step parameter towards maintaining the balance of a biped. While the ZMP has only limited control over the CoM during the step that is already being executed, the touch down of the swing foot determines the next pendulum base location and has a strong influence on the future trajectory of the CoM. Our concept to determine a suitable footstep location is based on prediction. Given the current CoM state $\mathbf{c}$, the already chosen step time T, and the bounded ZMP offset $\mathbf{Z}$, we can estimate the achievable end-of-step state $\mathbf{c}'$ that will be reached when taking the limited influence of the ZMP into account. We use the LIPM2D Predict Algorithm 8.2 for this, and compute

$$\mathbf{c}' = \textsc{Predict}(\mathbf{c}, \mathbf{Z}, T). \tag{8.28}$$

The step size adapts to a predicted state at the end of the step. The Zero Moment Point and external disturbances both influence the predicted end-of-step state.

The achievable state $\mathbf{c}'$ is a strong indicator of the state of balance in the near future. After a disturbance, the achievable state can significantly deviate from the nominal state $\mathbf{s}$. For example, if the robot is pushed from the back while it is walking forward, and the push is strong enough that it cannot be compensated by means of ZMP control, the future CoM state $\mathbf{c}'$ will inevitably overshoot with a higher than expected velocity. This situation is illustrated in Figure 8.6. The biped must react with a larger step size, otherwise its instability will increase during the next step. A simple way to think of it is that the biped has to adapt its step size to match the additional velocity it gained from the push.

In the following, we compute the sagittal and lateral coordinates of the footstep $\mathbf{S}' = (S'_x, S'_y)$, defined to be expressed relative to the coordinates of the predicted end-of-step state $\mathbf{c}' = (c'_x, \dot{c}'_x, c'_y, \dot{c}'_y)$. Due to their conceptually distinct behavior, we use different strategies for the sagittal and the lateral directions. To obtain the sagittal step coordinate S'_x, we determine the limit cycle trajectory that results in the same sagittal CoM velocity by the end of the step as the sagittal velocity $\dot{c}'_x$ of the predicted end-of-step state. We use the LIPM velocity predictor equation (8.4) with $(0, \dot{x}_0)$ as the initial state and solve $\dot{x}(\tau, 0, \dot{x}_0) = \dot{c}'_x$ for $\dot{x}_0$. We thereby determine the sagittal apex velocity $\dot{x}_0$ that would result in the end-of-step velocity $\dot{c}'_x$ after the half step time τ. We then use the LIPM location predictor equation (8.3) with the now known apex velocity $\dot{x}_0$ as the initial state $(0, \dot{x}_0)$ at the sagittal apex, and compute the end-of-step location $x(\tau, 0, \dot{x}_0)$ of the limit cycle. Since limit cycle steps are symmetrical, the end-of-step state coordinate $x(\tau, 0, \dot{x}_0)$ is equal to the sagittal footstep coordinate S'_x. All of the operations outlined above yield the sagittal step size

$$S'_x = \frac{\dot{c}'^2_x}{C} \tanh(C\tau). \tag{8.29}$$

The lateral step size S'_y is computed such that the CoM will pass the apex of the next step at a distance α. A simple way to derive the formula is using the constant orbital energy (8.9). The orbital energy $E(S_y, \dot{c}'_y)$ right after the support exchange should equal the constant energy level of the lateral step apex $E(\alpha, 0)$. Solving the equation $E(S'_y, \dot{c}'_y) = E(\alpha, 0)$ for S'_y yields the lateral step size

$$S'_y = \lambda \sqrt{\frac{\dot{c}'^2_y}{C^2} + \alpha^2}, \tag{8.30}$$

where $\lambda \in \{-1, 1\}$ is the sign of the support leg prior to the step.

The transformation of $\mathbf{S}'$ into the foot-to-foot step vector $\mathbf{S}$ is trivially given by

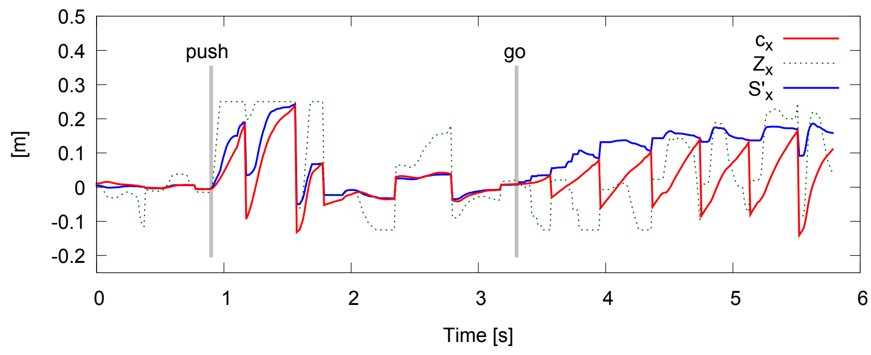
$$\mathbf{S} = \left(c'_x + S'_x,\ c'_y + S'_y,\ \check{S}_\psi\right). \tag{8.31}$$

The rotational step size $\check{S}_\psi$ is simply passed through the balance control without modification. We assume that the rotational gait control component does not have a significant influence on the sagittal and lateral balance. The yaw motion of the legs transforms sagittal into lateral motion and vice versa and the balance controller should be able to handle a small amount of rotation per step automatically. The same assumption was made by Kajita et al. [2003].

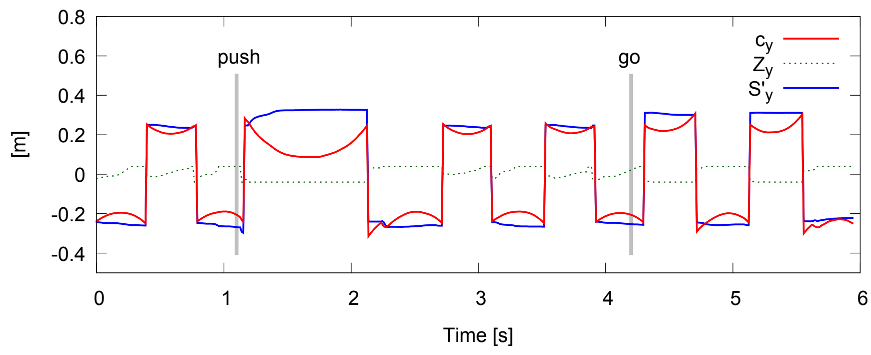
The Zero Moment Point is not passed on to the motion generator.

The now complete step parameters $(\mathbf{S}, T)$ are passed on to the Motion Generator module to command the robot to step to the computed location at the right time. A noteworthy detail about our framework is that neither the ZMP, nor the CoM trajectory, is passed on to the Motion Generator. The step size $\mathbf{S}$ and the step time T are implicit results of the computed ZMP offset $\mathbf{Z}$. The physical execution of the commanded step should place the ZMP location at least roughly in the right location under the foot, even without a means to explicitly enforce the physical location of the ZMP. Both real and simulated robots have successfully been controlled by the Capture Step Framework, and have shown robust bipedal walking capabilities, even though the ZMP control is included only in the theoretical model. Concerning the CoM trajectory, we argue that it is beneficial not to force a CoM trajectory upon the robot that was generated by a low-dimensional model. The Motion Generator should be allowed to produce high-dimensional whole-body stepping motions as freely as possible, as long as the commanded foot placements are met at the right time.

The presentation of the theoretical aspects of the analytic balance controller is now complete. In the next section, we guide the reader through an experiment that highlights the most significant signals of the analytic footstep controller, and we present the results of a stability analysis.



(a) Sagittal



(b) Lateral

Figure 8.8: Center of Mass, Zero Moment Point and footstep coordinate data recorded with Dynaped during a (a) sagittal and (b) lateral push and walk experiment. The vertical bars indicate the moments when the robot was pushed (push), and then commanded to walk forward and to the left, respectively (go).

8.5 EXPERIMENTS

8.5.1 *Walk and Push*

To demonstrate the analytic footstep controller in operation, we performed a push and walk experiment with the humanoid robot Dynaped. Figure 8.8a shows the relevant data streams of a sagittal experiment, and 8.8b shows a lateral experiment. In both experiments, the robot is first pushed while walking in place. Some time later, it is commanded to walk forward or to the side, respectively.

In the sagittal direction, the push occurs at time 0.9 s towards the back of the robot while it was walking on the spot. The CoM trajectory indicates how the robot suddenly starts to step forward after the push. The robot takes several steps, during which the sagittal ZMP is shifted to its forward limit of 25 cm. The step size subsequently decreases and the robot comes to a halt. Then, it is commanded to walk forward after 3.2 s. The ZMP shifts to the “heels”, and the robot accelerates forward by increasing its step size step by step.

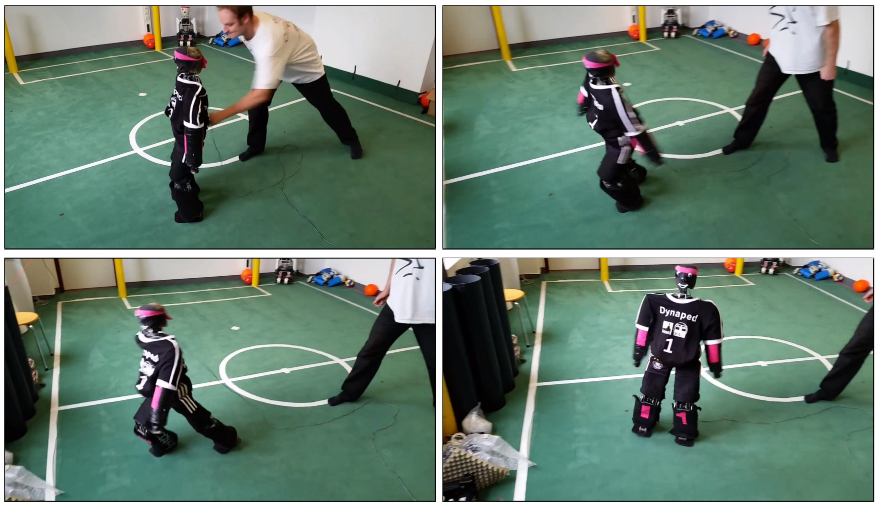


Figure 8.9: Extreme push recovery with bipedal robot Dynaped.

In the lateral direction, the push occured approximately at 1.1 s from the side. The disturbance in the CoM trajectory is clearly visible. The recovery step takes approximately than one second to execute. The lateral ZMP is at its limit during this time. At the same time, the lateral step size increases to counteract the disturbance during the next step. The robot returns to its nominal oscillation amplitude after one step, before it is commanded to walk to the side at 4.2 s.

8.5.2 *Technology Demonstration*

The robust omnidirectional walking skills of Dynaped were shown in a technology demonstration at the RoboCup German Open in 2014.

We gave a technology demonstration at the RoboCup German Open robot soccer event in Magdeburg in April 2014, where the analytic footstep controller was shown during one hour long public demonstrations several times over the course of two days. A short demonstration was also featured during the half-time of the finals in the KidSize class. Heise Online mentioned the technology demonstration in an article[1] about the event. Video [2] shows a five minute long compilation of recorded scenes, where bipedal robot Dynaped demonstrates reliable and controllable walking skills with disturbance rejection capabilities. In this video, Dynaped was not only disturbed by pushes during walking, but also by placing a hand under its feet, and by collisions with a static obstacle. In a further robot demonstration at the French-German-Japanese Conference on Humanoid and Legged Robots in 2014, a video [6] was recorded of Dynaped walking outside on a cobblestone surface. In the experiment shown in video [5], we mounted feet of human-like proportions on Dynaped that were smaller than the existing ones, and after refitting the parameters, we reproduced omnidirectional walking capabilities of similar quality as before. We also managed to produce a few quite extreme

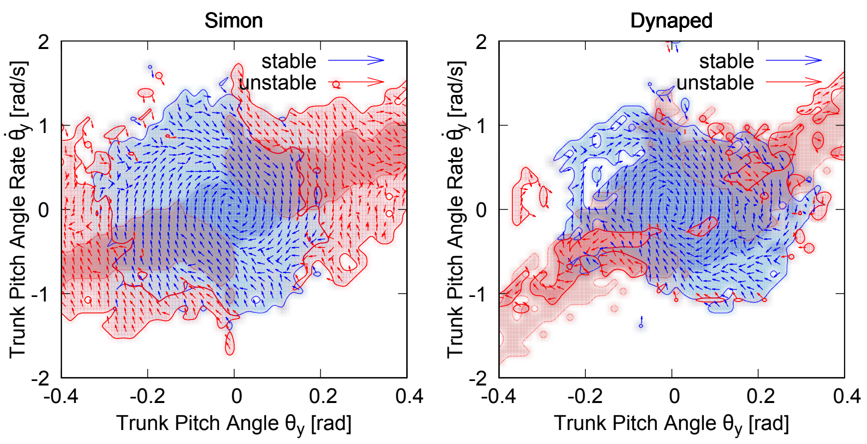
1 http://goo.gl/CN8ZS3

cases of push recovery, as shown towards the end of video [5] and in Figure 8.9.

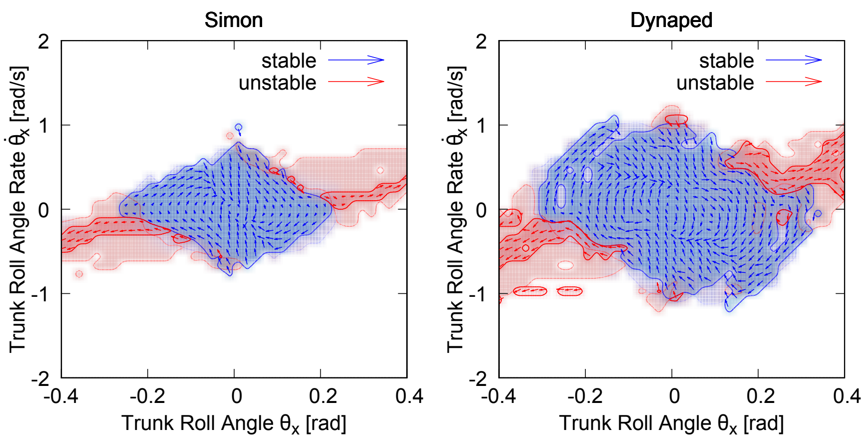
8.5.3 *Push Recovery*

In a systematic push recovery experiment, we explored the stability of the analytic capture step controller in simulation and with a real robot. In the same manner as in the exhaustive push experiment with the open-loop CPG in Section 6.5, we subjected Simon and Dynaped to a large number of pushes from the front and the back, and from the left and the right, to map out the stable regions in the sagittal and the lateral trunk angle phase spaces. During this testing the analytic footstep controller was active. Simon was pushed 300 times in each direction and fell 24 times in the sagittal direction and 57 times in the lateral direction. Dynaped was pushed 96 times in the lateral direction, out of which 26 cases resulted in a fall. The sagittal experiment had to be aborted after 85 pushes and 22 falls, when Dynaped suffered damage to its IMU. The proportion of pushes that led to a fall does not characterize the stability very well. It is rather an indicator for the fact that the controller was tested in a range of disturbances that includes pushes strong enough to push the robots beyond their limits. Since less data could be gained from the real robot, and as it is difficult to produce a well distributed set of pushes by hand, the plots showing the results of the real robot experiment are more noisy, but show the same qualitative result as the simulated experiment.

Figure 8.10 shows vector fields and color coded regions in the sagittal and the lateral trunk angle phase spaces. For a detailed description of how the vector fields and the regions are constructed, please refer to Section 6.5. The color blue indicates the stable region, where the controller is able to reliably stabilize the robot. When the trunk angle is pushed into the red region, in most cases the controller is not able to restore the balance of the robot. The regions of the open-loop CPG experiment are shown in darker colors in the background for a direct comparison of their size. In the sagittal direction, the analytic controller clearly increases the size of the stable regions. This means that the analytic controller is able to restore balance in cases where the robot has a high enough tilt, or angular velocity, for which it would normally tip over the edge of a foot and fall, if no corrective action were taken. In the lateral direction, the size of the stable region is not increased. The reason for this is that the analytic controller is not able to handle the situation where the robot tips over the support leg. Thus, the analytic controller operates in the same range of trunk roll angles, as the open-loop controller does. The vector fields are most interesting when comparing with the results of the same experiment that was performed with the open-loop CPG shown in Figure 6.11. For the sagittal direction, the vectors in the red regions of the open-loop



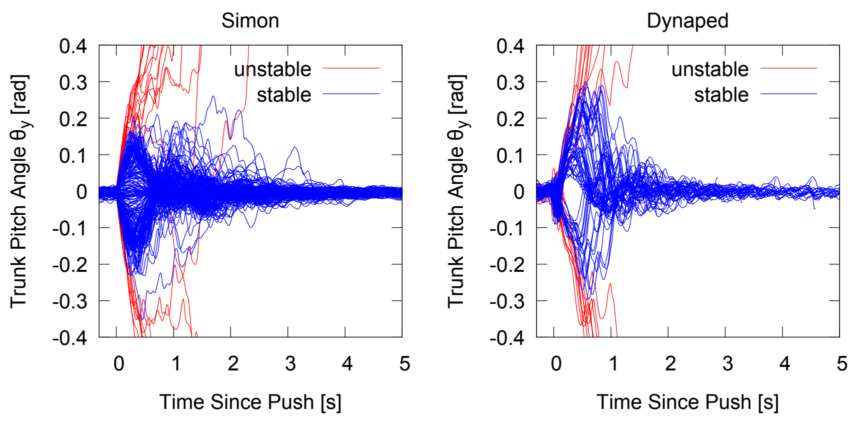
(a) Sagittal



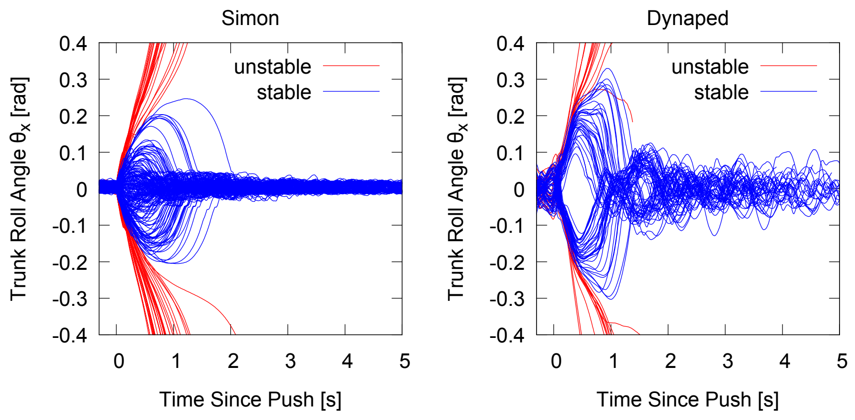
(b) Lateral

Figure 8.10: Vector fields of (a) the sagittal (pitch) and (b) the lateral (roll) phase spaces of the trunk angle with the analytic footstep controller. The simulated robot Simon and the real robot Dynaped were subjected to pushes of randomly varying strength from the front and the back to explore the sagittal stability and from the left and the right to map out the lateral stability. The vectors point in the direction the trunk angle is moving in on average in this region of the phase space. The length of the arrows hints at the angular velocity. The stable region, where the controller reliably rejects disturbances, is colored in blue. States that most certainly lead to a fall are marked in red.

experiment point away from the stable region. In the experiment with the analytic controller, these vectors are tangential to the stable region, indicating that the controller is still making a reasonable attempt to stabilize the robot. In the lateral direction, the vectors generated by the analytic controller are much more directed towards the central region of the phase space than the vectors produced by the open-loop controller.



(a) Sagittal



(b) Lateral

Figure 8.11: Plots of (a) the pitch (sagittal) and (b) the roll (lateral) trunk angle trajectories synchronized by the moment of the push.

The analytic controller increases the stable region in the sagittal phase space, and improves the lateral balance by resolving cases of self-disturbance, and by accelerating the return to a nominal oscillation.

In Figure 8.11 trunk angle trajectories are plotted that have been synchronized by the moment of the push. Trajectories that resulted in a fall are marked in red. Trajectories that were successfully balanced are marked in blue. In the simulated sagittal experiment, a few cases of overcompensation can be observed, where for example, the robot was pushed backwards, the controller overcompensated with a too large step, and the robot fell forward in the end. In comparison with the trunk angle trajectories from the open-loop experiment, shown in Figure 6.12, the following observations can be made. In the sagittal direction, a significantly higher push impact can be absorbed, as indicated by the peaks of the trunk pitch angle trajectories shortly after the push. In the lateral direction, the peaks are approximately the same size. However, the open-loop experiment shows cases of self-disturbance, where the initial push was not the cause of the fall, but an inappropriate step at a later time. Additionally, even when the robot did not fall after the push, it takes a long time to return to its nominal amplitude of oscillation. This is in particular clear to see in the trunk roll angle trajectories of Dynaped. With the analytic controller enabled, cases of self-disturbance are completely eliminated and Simon and Dynaped return to their nominal trajectories in one or two steps.

8.6 DISCUSSION

In this chapter, we have presented an analytic version of a footstep controller that tracks a commanded step size while maintaining balance. A low-dimensional point mass model with linear inverted pendulum model dynamics was used to represent the CoM trajectory and to compute the ZMP, step timing, and step size parameters that fulfill the footstep control task. The controllability and the robustness of the bipedal walk was demonstrated in videos and through systematic push recovery experiments.

The update frequency of the gait generation process is 83.3 Hz. Each iteration of the control software requires 0.12 ms to compute on a single 1.3 GHz core for the entire control loop including state estimation, balance control, and motion generation. The execution time is spent almost entirely on the computation of closed-form formulae. This low computation time would allow for the application of the analytic controller at a much higher frequency. The bottleneck in our setup is the hardware communication layer, which is not able to transport the amount of data in the timely fashion that would be required at a higher frequency.

The key of the highly responsive analytic controller is the computation of the step size based on a prediction of the future end-of-step state of the low-dimensional point mass. When either the control input, or a disturbance, alters the course of the CoM mid-step, the predicted end-of-step state changes too, and the balance controller can immediately respond by suggesting a new footstep location. In fact, the analytic controller performs the same computation in every cycle based on an almost arbitrary constellation of the CoM and target state, and does not distinguish between stable and unstable states. This eliminates the need to sense and categorize the type of a disturbance, and does not require an estimate of its magnitude. The controller is automatically robust to a number of disturbances of different nature.

A remarkable detail that arises in our particular implementation is that since we do not have the means to enforce or to measure the ZMP, it is not passed on to the motion generation layer. The physical location of the ZMP arises freely from execution of the motion. Despite a high amount of latency and imprecise actuation, commanding the step size and the step time alone is sufficient to control a bipedal robot to an extent where push recovery is possible.

9 LEARNED FOOTSTEP CONTROL

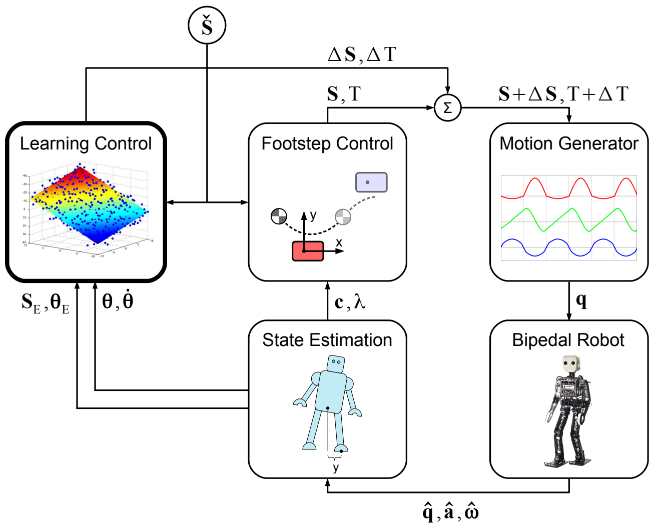


Figure 9.1: The Learning Control component (top left) is embedded into the Capture Step Framework. The Learning Control contributes the step size and timing offsets ΔS and ΔT, which are added to the outputs of the analytic Footstep Control. The Learning Control receives two data streams from the State Estimation. The trunk angle θ and angular velocity $\dot{\theta}$ estimates are used for continuous balance control in every iteration of the control loop. The step size S_E and trunk angle θ_E at the end of the step are used to train the Learning Control.

Using machine learning to train a capable gait controller with the hardware in the loop is thought to be a promising alternative to engineered design. However, model-free learning of a high-dimensional motion signal for the motors of a humanoid robot in a way that a robust and controllable walk is achieved, is intractable. With the help of a motion skeleton and a fitness function that guides the learning process towards upright walking, evolutionary algorithms and reinforcement learning methods can be successful in simulation. When a real robot is in the loop, the feasible number of trials and the risk of damaging the hardware become limiting factors, and the problem of learning to walk needs to be approached in a new way.

Machine learning with real hardware is difficult due to the limited number of feasible repetitions.

Flight animals learn how to balance their gait shortly after birth based on a genetic disposition to step in the right direction. This biological example suggests that initialization with a hard-wired motion pattern—and a rough concept of balance—are key factors for the learning of balance-critical locomotion skills in an online learning

Biological examples of learning to walk often start with a genetically initialized concept of motion and balance.

Initialization with the Central Pattern Generator and the analytic footstep controller allows learning with a robot that can already maintain balance to some extent.

setting. We follow this paradigm and bootstrap the learning process with our CPG, which enables the robot to execute targeted stepping motions out of the box. The CPG hides the full complexity of the walking motion and reduces the learning task to the learning of Cartesian footstep coordinates and the timing of steps, as opposed to learning whole-body control. The initialization with a rough concept of balance is twofold. On the one hand, the learning algorithm can coexist with the previously introduced analytic footstep controller. The analytic controller initializes a controller response for the entire state space and provides a starting point where the robot can already walk and maintain balance to some extent. This way, the exploration can start from a stable state, and the first steps do not have to be learned from the experience of falling. On the other hand, we use a pendulum-cart as a simple physical model to drive the learning process. Based on feedback of the errors the robot makes during walking, we infer sagittal step size modifications using an estimated gradient that we gain from the physical model. The gradient boosts the rate of learning to a level of performance where the experience of a few steps can be sufficient to successfully learn how to recover from a push, even when the analytic initialization is not present.

Despite the fact that our method is limited to the learning of the sagittal step size, we choose to set our approach into the context of a bigger picture. We introduce our vision of learning a footstep controller in greater generality, and present our method of learning the sagittal step size as a part of the outlined concept.

The learning footstep controller learns an offset to the analytic controller.

Our online learning scheme is embedded into the Learning Control component of the Capture Step Framework, illustrated in Figure 9.1. The idea of the Learning Control is to be able to learn a step size offset $\Delta\mathbf{S}$ and a timing offset ΔT that are to be added to the output of the analyic Footstep Control. The modified step size output $\mathbf{S} + \Delta\mathbf{S}$ and the modified step time output $T + \Delta T$ are passed on to the Motion Generator module as the commanded step parameters. The Learning Control has access to the commanded step size $\check{\mathbf{S}}$ and pursues the same task as the analytic Footstep Control—tracking the reference step size while maintaining balance. The Learning Control receives input from two data streams. The trunk angle θ, and its angular velocity $\dot{\theta}$, drive the footstep control function of the Learning Control, which is executed in every iteration of the control loop. This stands in contrast to the analytic controller, which is based on the CoM state $\mathbf{c}$ and the support leg sign λ. The second data stream delivers the step size $\mathbf{S}_E$ and the trunk angle θ_E at the end of the step, both measured in the moment of the support exchange. These quantities are used for the training of the sagittal step size.

In the following, we formally introduce our concept of learning a footstep control function online, and descend into the details of the completed component—learning of the sagittal step size. Finally, we present experiments and discuss the results.

9.1 MACHINE LEARNING FRAMEWORK

The Learning Control is formally a footstep control function

$$(\boldsymbol{\Delta S}, \Delta T) = \mathcal{F}(\check{\mathbf{S}}, \theta, \dot{\theta}, \lambda), \tag{9.1}$$

where $\boldsymbol{\Delta S} = (\Delta S_x, \Delta S_y, \Delta S_\psi)$ is the step size offset, ΔT is the step timing offset, $\check{\mathbf{S}} = (\check{S}_x, \check{S}_y, \check{S}_\psi)$ is the commanded step size, $\theta = (\theta_x, \theta_y)$ is the trunk angle, $\dot{\theta} = (\dot{\theta}_x, \dot{\theta}_y)$ is the angular velocity of the trunk, and λ is the support leg sign. The approach suggested in this thesis to train the footstep control function online hinges on a strong reduction of the complexity of the learning task. The most significant reduction of complexity, by learning only the control parameters of the CPG, is already reflected by the footstep control equation (9.1). While the number of input and output dimensions of the footstep control function $\mathcal{F}$ is much lower than what would be needed for whole-body control, we simplify the task further by decomposing $\mathcal{F}$ into independent control functions

The decomposition of the footstep controller into independent control functions simplifies the learning task.

$$\mathcal{F}(\check{\mathbf{S}}, \theta, \dot{\theta}, \lambda) = \begin{bmatrix} \Delta S_x \\ \Delta S_y \\ \Delta S_\psi \\ \Delta T \end{bmatrix} = \begin{bmatrix} \mathcal{F}_x(\check{S}_x, \theta_y, \dot{\theta}_y) \\ \mathcal{F}_y(\check{S}_y, \theta_x, \dot{\theta}_x, \lambda) \\ \mathcal{F}_\psi(\check{S}_\psi) \\ \mathcal{F}_T(\theta_x, \dot{\theta}_x, \lambda) \end{bmatrix}, \tag{9.2}$$

where ΔS_x is the sagittal step size offset, ΔS_y is the lateral step size offset, ΔS_ψ is the rotational step size offset, and ΔT is the offset of the the step time. This decomposition is motivated by the dimensional splitting of the walking motion into the sagittal and lateral directions, as introduced in Chapter 5, which suggests that we could learn separate controllers for the sagittal and the lateral directions, and allocate the control of the step time to the lateral direction. We further assume that the rotational step size controller $\mathcal{F}_\psi(\check{S}_\psi)$ does not need to be concerned with balance, and learns only to increase its precision with regard to the desired step yaw $\check{S}_\psi$.

We propose the trunk angle, the angular velocity of the trunk, and step size measurements to be used as sources of information for the operation and training of the learned footstep control function $\mathcal{F}$. These quantities can be reliably estimated and are not prone to excessive noise, as opposed to for example the CoM velocity, which has to be determined via numerical differentiation. While the trunk angle and the angular velocity of the trunk are strong indicators of the state of balance, the measured step size is a direct estimate of the reference tracking error. Specifically for training, we measure these quantities in the instant of the touch-down of the swing foot and obtain the

The trunk angle and the step size are reliable sources of information that express the state of balance and the reference tracking error.

end-of-step trunk angle $\theta_E = (\theta_{Ex}, \theta_{Ey})$, and the step size estimate $S_E = (S_{Ex}, S_{Ey}, S_{E\psi})$. The step size estimate is gained from the tilted whole-body pose reconstruction through computation of the transformation from the ground projection of the support foot to the ground projection of the swing foot. The detection of the moment of the support exchange implies the notion of the support leg sign λ, which can also be estimated reliably from the pose reconstruction as long as the floor is horizontal and flat. We use the step size estimate S_E and the commanded step size $\check{S}$ to compute the step size error

$$\bar{S} = S_E - \check{S}. \tag{9.3}$$

A learning gradient can be gained from simple physical models and can be used to boost the learning performance.

To facilitate fast learning, we expect to find a simple, possibly conceptually different model for each component of the footstep control function $\mathcal{F}$ for the purpose of computing an approximate gradient. We define the gradient function

$$\mathcal{G}(\bar{S}, \theta_E) = \begin{bmatrix} \mathcal{G}_x(\bar{S}_x, \theta_{Ey}) \\ \mathcal{G}_y \\ \mathcal{G}_\psi \\ \mathcal{G}_T \end{bmatrix} \tag{9.4}$$

to be a function of the end-of-step trunk angle θ_E and the step size error $\bar{S}$. The gradient function $\mathcal{G}$ is decomposed into independent gradients for the elementary functions of the footstep control function $\mathcal{F}$. The gradient can be used for an efficient online modification of the step size coordinates and the step timing, respectively. For the learning of the sagittal step size, we found the pendulum-cart model to be a powerful model for the implementation of a suitable gradient function $\mathcal{G}_x(\bar{S}_x, \theta_{Ey})$. In earlier work [Missura et al., 2014], the LIPM was used to deduce an approximate gradient for the learning of the lateral step size. However, this gradient was based on the CoM state rather than the trunk angle, and it was suitable for push recovery learning, but not for reference tracking. Thus, we choose not to include it into our framework, and use it only to support the assumption that a simple model can be found for the learning of the lateral step size. Under the assumption that the yaw step size S_ψ has only a negligible effect on balance, the yaw step size gradient can possibly be as simple as the difference between the target and the measured yaw step size. The components $\mathcal{G}_y$, $\mathcal{G}_\psi$, and $\mathcal{G}_T$ of the gradient function have not been implemented and cannot be further specified.

The learned footstep control function is represented by a function approximator that is trained with the learning gradient.

The footstep control function $\mathcal{F}$ is represented by a function approximator, which is initialized with zero output for all states. The function approximator is updated in a specific way that arises from having gradient information at hand. We query the function approximator at the location $(\check{S}, \theta, \dot{\theta})$ of the input space that we wish to update, add the gradient $\mathcal{G}(\bar{S}, \theta_E)$ to the returned value with consideration of a learning rate, and present the new value to the function approximator as the desired output at this location. The update function at the point $(\check{S}, \theta, \dot{\theta})$, using the gradient $\mathcal{G}(\bar{S}, \theta_E)$, is given by

$$\mathcal{F}(\check{S}, \theta, \dot{\theta}) := \mathcal{F}(\check{S}, \theta, \dot{\theta}) + \eta\, \mathcal{G}(\bar{S}, \theta_E), \tag{9.5}$$

where $\eta > 0$ is a learning rate. This method of incremental learning was introduced in earlier work, where it was used for the learning of the lateral step size in a push recovery setting [Missura et al., 2014]. The method has remained unchanged for the learning of the sagittal step size. This is no surprise, as the update method is not specific to the learning task, but only to the setting of gradient based training of a function approximator in an online fashion.

We use an implementation of the LWPR algorithm for the realization of an online capable function approximator.

The online learning setting demands specific capabilities of the function approximator implementation. The main control loop queries the footstep control function at a high frequency—typically 100 Hz—to drive the walking motion. A function approximator used in this context has to deliver a time-critical response, even when it is being updated with new data. Neither the response time nor the memory consumption of the function approximator should degrade with the ever increasing amount of seen data, otherwise the learning process will eventually have to terminate. Gaussian processes [Rasmussen and Williams, 2006], regression trees [Breiman et al., 1984], and random forests [Breiman, 2001], all degrade when used in an incremental learning setting. The LWPR algorithm [Vijayakumar et al., 2005] has been specifically designed for incremental learning. It represents a function with a bounded number of locally linear kernels, such that the training data can eventually be discarded. Thus, the memory consumption, update times, and recall times are bounded. We use an open-source implementation[1] of the LWPR algorithm that fully satisfies our requirements.

9.2 LEARNING THE SAGITTAL STEP SIZE

For the training of the sagittal step size control function $\mathcal{F}_x(\check{S}_x, \theta_y, \dot{\theta}_y)$, we use the sagittal component of the reference step size tracking error $\bar{S}_x = S_{Ex} - \check{S}_x$, and the end-of-step trunk pitch angle θ_{Ey}. From these two quantities, we compute the gradient $\mathcal{G}_x(\bar{S}_x, \theta_{Ey})$ and use it to improve the sagittal step size for the states that were encountered during the step that resulted in the above measurements.

The pendulum-cart model resembles the angular dynamics of a biped.

To derive a suitable modification of the step size that attempts to maintain an upright posture of $\theta_y = 0$, we borrow a concept from the pendulum-cart model illustrated in Figure 9.2. The simplest controller that manages to balance the pendulum on the cart when the pendulum angle $\phi \approx 0$, is a proportional controller $\ddot{x} = k\phi$ for some gain k. Here, the acceleration $\ddot{x}$ of the cart is a function of its angle ϕ.

1 http://wcms.inf.ed.ac.uk/ipab/slmc/research/software-lwpr

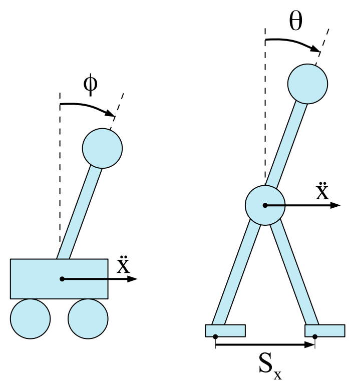


Figure 9.2: The pendulum-cart model (left) resembles the angular dynamics of a biped (right). When the cart accelerates, the pendulum angle is accelerated in the negative direction. A biped accelerates by increasing its step size and can counteract undesired angular momentum the same way as the pendulum-cart.

When the biped is tilted "forward" by the end of the step, it should have made a longer step.

A biped is not a cart, but it can accelerate its center of mass by increasing or decreasing its step size. We can translate the proportional controller from the pendulum-cart to a bipedal setting by converting acceleration to a step size modification.We thereby obtain a rough approximation of a balancing step controller

$$\Delta S_x \approx k\,\theta_{Ey}, \tag{9.6}$$

which estimates a sagittal step size modification ΔS_x to control the trunk pitch angle. An example is shown in Figure 9.2. If at the end of a step the trunk pitch angle θ_{Ey} is positive, i. e. the robot is rotated "forward", the robot needs to take a larger step next time in the same situation in order to end up with a more upright posture.

The right place to step to maintain balance and the desired footstep location are not necessarily the same.

As the right place to step to counteract undesired angular momentum does not necessarily coincide with the desired step size, we face an ill-posed problem. A trade-off must be found between stepping onto a desired location and avoiding a fall. We combine the control law we derived from the cart-pendulum model (9.6), and the footstep error $\bar{S}_x$, into the gradient function

$$\mathcal{G}_x(\bar{S}_x, \theta_{Ey}) = \theta_{Ey} - p_\theta \tanh(p_S \bar{S}_x). \tag{9.7}$$

The characteristic saturation of the hyperbolic tangent function limits the influence of the step size error $\bar{S}_x$ to a configurable bound of p_θ for two specific purposes. The parameterized saturation makes sure that the robot learns to track the reference step size carefully in order to avoid sudden changes of the CPG activation signal that are likely to cause instability, and, critical inclinations of $\|\theta_y\| \gg p_\theta$ dominate the gradient, ensuring balance takes priority over reference tracking when a fall is imminent. p_S is a weight to fine-tune the influence of the step size error within the permitted bounds. Throughout our experiments, we used $p_\theta = 0.15$ and $p_S = 30$. Note that the gain k has been dropped from the gradient equation, because it is absorbed by the learning rate that we multiply the gradient with when we apply the update rule in equation (9.8) below.

We represent the sagittal step size control function $\mathcal{F}_x(\check{S}_x, \theta_{y_i}, \dot{\theta}_{yi})$ with a dedicated function approximator. At the end of each step, we train the function approximator with the update rule

$$\mathcal{F}_x(\check{S}_x, \theta_{y_i}, \dot{\theta}_{yi}) := \mathcal{F}_x(\check{S}_x, \theta_{y_i}, \dot{\theta}_{yi}) + \eta\, \mathcal{G}_x(\bar{S}_x, \theta_{Ey}), \forall i \in I, \quad (9.8)$$

where $\eta = 2.0$ is the learning rate, I is an index set, and $\{\theta_{y_i}, \dot{\theta}_{yi}\}, i \in I$ is the set of trunk pitch angles and angular velocities that were measured during the step. In words, we query the function approximator at the locations that were seen during the step, add the gradient to the returned values, and present the results as the new desired outputs to of function approximator at the respective locations.

9.3 EXPERIMENTS

To evaluate and demonstrate isolated features of our learning framework, we performed a series of experiments in simulation and with a real robot. In all of the following experiments, the Learning Control learned online during the experiment. It was not pre-trained and it was operational during the entire evaluation time. Although the experiments are focused on learning the sagittal step size, the motion of the robot was not restricted in any way and the analytic controller was fully operational.

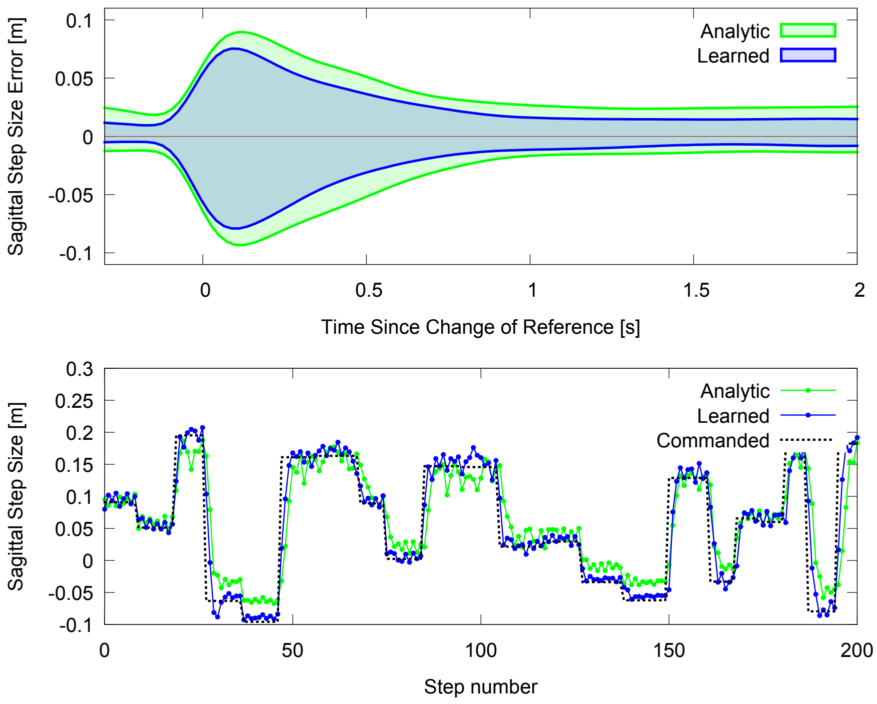
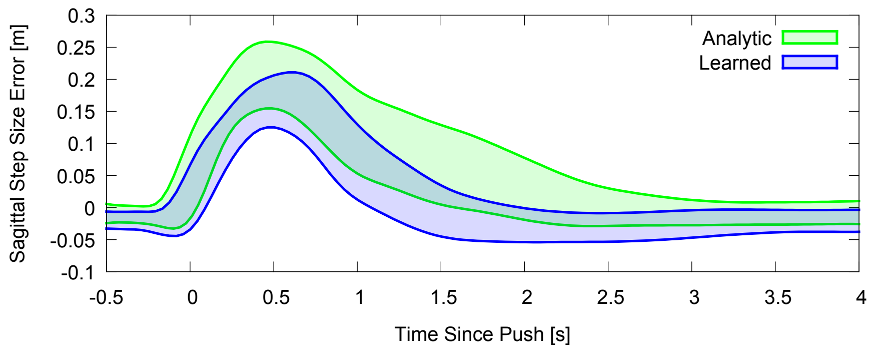


Figure 9.3: *Top:* In an experiment with random changes of the commanded sagittal step size, the standard deviation of the step size error decreases faster with learning, than without. *Bottom:* A time series of the commanded step size and the step sizes produced by the analytic and the learned controllers.

9.3.1 *Evaluation of Reference Tracking*

The reference tracking capabilities improved with learning.

In this experiment, we evaluate the ability of the controller to track a reference step size in simulation with Simon. We compare the analytic footstep controller on its own, and the analytic controller together with the learning controller that was trained during the experiment. We sample the commanded step size $\check{S}_x$ from a range $[-10, 20]$ cm, and keep it constant for 4 to 8 seconds. We observe how quickly both controller versions can adapt the step size to the correct value. To preserve comparability, the same random step size sequence was presented to both controller versions. Figure 9.3 shows statistical data averaged over 1000 steps. The moments of the reference step size changes are synchronized at zero seconds. We observe the mean and standard deviation of the step size error. Since the reference step size was uniformly sampled, the mean is near zero. With learning, the standard deviation of the step size error decreased at all points in time. This means that the learned controller not only follows the reference faster than the capture step controller alone, but it also learned to step onto the right location altogether more precisely. In the bottom plot in Figure 9.3, we show a time series extract of the commanded and the measured sagittal step sizes.



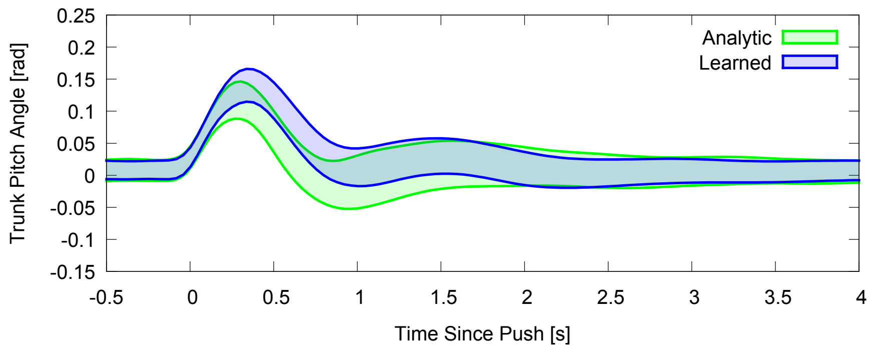


Figure 9.4: *Top:* After the robot was pushed, the sagittal step size increases when the robot steps forward in order to prevent a fall. When the learning controller enabled, the robot is able to return faster to the commanded step size. *Bottom:* The learning controller allows a larger trunk angle within the allowed margin of 0.15 radians in order to better obey the commanded step size.

9.3.2 *Evaluation of Disturbance Rejection*

In this experiment performed with the simulated robot Simon, we demonstrate the ability of the controller to return to a commanded step size after a sagittal disturbance. We command the robot to walk forward with a fixed sagittal step size of 20 cm. While the robot is walking, we push it 400 times onto the back with impulses of magnitude 8 Ns. The pushes are triggered at random times in order to avoid hitting the robot repeatedly in the same motion phase. Synchronized by the moment of the push, Figure 9.4 shows how the step size (shown on the top) and the trunk pitch angle (shown on the bottom) return to their reference values. The learning component reduces the time it takes for the robot to return to the commanded step size. The fact that the trunk pitch angle appears to have increased after learning might be a surprise at first, but seeing as we set the tolerated trunk pitch angle parameter p_θ to 0.15 radians, the learned controller utilized this allowable margin to sacrifice a small amount of balance in order to better track the commanded step size closer.

The learning controller learned to sacrifice a small amount of balance in order to better track the desired step size.

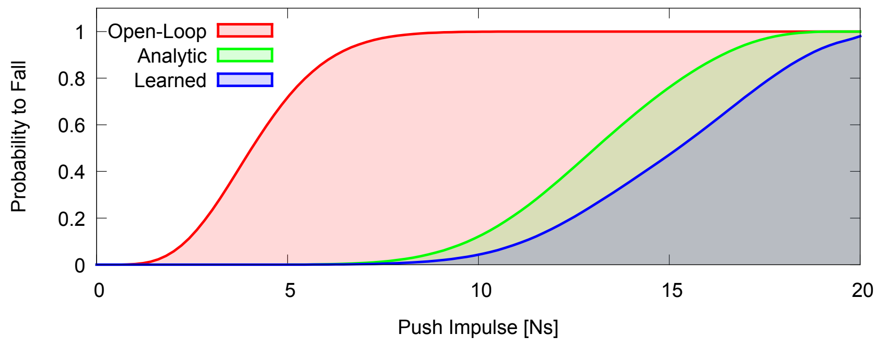


Figure 9.5: Probability to fall of an open-loop, analytic, and a learned controller with respect to varying push impulses from the back.

9.3.3 *Evaluation of Stability*

The learned controller learned to utilize the planar steps of the motion generator more efficiently than the analytic controller and can absorb a stronger push.

In this simulated experiment we aim to evaluate whether the learning component is able to improve the sagittal stability. We apply 400 randomly timed push impacts to a robot walking in place, with magnitudes sampled from a range of $[0, 20]$ Ns. The pushes are directed in the forward direction and force the robot to make forward steps in order to avoid falling. By counting falls and pushes, we estimate the probability of a fall depending on the magnitude of the disturbance. In addition to the Analytic and the Learned footstep controllers, in this experiment we also included an open-loop controller that walks in place with a fixed frequency and does not react to the pushes. The results are shown in Figure 9.5. The analytic controller significantly increases the push resistance compared to what the robot can absorb passively. The learned controller increases the stability even further. Both controllers face the limitation that the motion generator and the footstep controller do not take the inclination of the floor with respect to the robot into account. Strong pushes force the robot to tip forward into an oblique pose, where the large forward step that would be necessary to counteract the push results in a kick into the floor. The learned controller can compensate this issue to some degree by learning to increase the size of the recovery step even more, but it quickly reaches the limitations of the robot, and cannot significantly improve the push recovery performance as compared to the analytic controller. At this point it is clear that in order to be able to absorb even stronger pushes, the gait control framework has to be upgraded to anticipate the tilt of the robot at the end of the step, and to adjust the motion pattern accordingly.

9.3.4 *Push Recovery Learning with a real Robot*

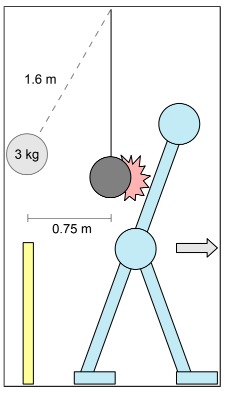


Figure 9.6: Experimental setup for push recovery learning.

We evaluated the potency of the sagittal step size learning method in a push recovery experiment with the humanoid robot Copedo. The experimental setup is illustrated in Figure 9.6. We swing a 3 kg mass attached to a 1.6 m rope onto the back of the robot from a distance of 0.75 m. The mass is pulled back and released by hand. A yellow pole marks the starting point to aid manual repeatability. The robot is positioned at the spot where the rope reaches the vertical position. The mass of the robot is 8 kg. The experiment has been set up in a way that the impact of the mass on the back of the robot generates a significant push impulse that requires active control effort as a response. The passive stability of the robot is not enough absorb the impact.

We command the robot to walk in place by setting $\check{S}_x = 0$. This time, the step size output of the analytic controller was suppressed and the learning controller had to learn the concept of the sagittal step size on its own. Only the step timing component of the analytic controller was active in order to prevent the accumulation of lateral instability. The learning controller is initialized with a zero step size and trained online during the experiment. The impact is rather strong and the controller has to learn to step forward in order to cope with the push, but as soon as balance is restored, the robot should stop walking forwards and come to a halt due to the commanded step size of zero.

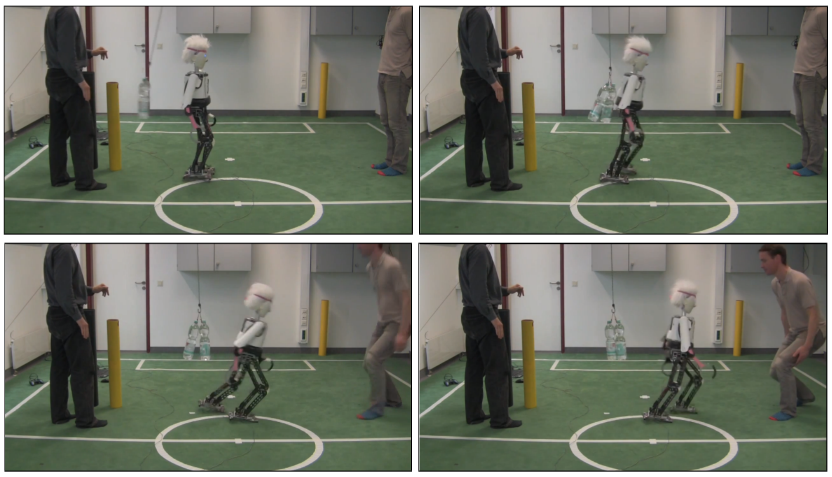


Figure 9.7: Humanoid robot Copedo regains its balance with a learned push recovery controller after a strong push from the back.

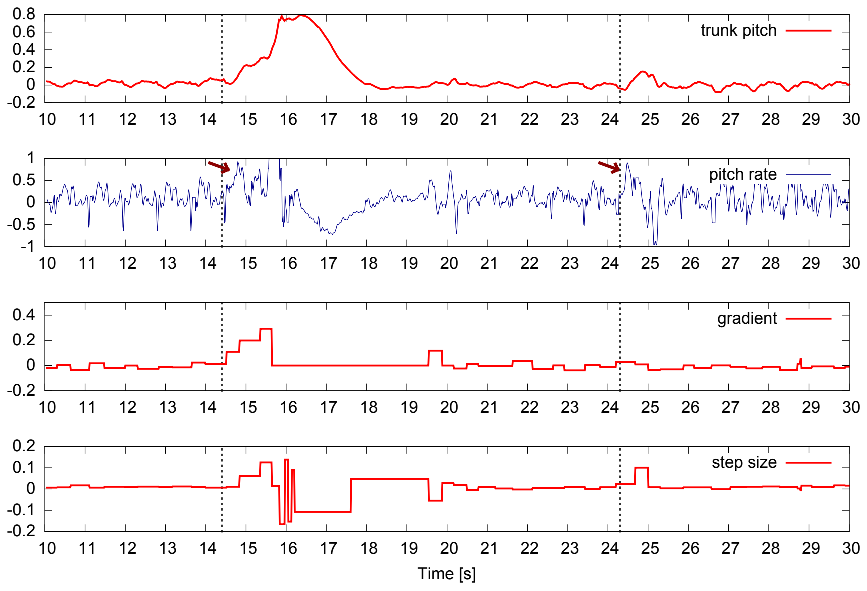


Figure 9.8: Data recorded during the sagittal push recovery experiment.

Even without initialization, robot Copedo learns to absorb a strong push from a few steps during falling.

A video of this experiment [7] shows uncut recordings of how Copedo successfully learns to absorb the impact. Due to the absence of the analytic controller, the learned controller cannot respond to the first push and Copedo falls forward. However, the controller learns from the unsuccessful steps during the fall and is able to successfully absorb the second push. When observing the attempted steps following the first push in slow motion, one can see with the naked eye how the controller is learning and attempting to increase the step size. Photographs of the experiment are shown in Figure 9.7. Note that it is not necessary for the robot to fall in order to train the sagittal balance controller. Light pushes that drive the robot to the limit of its balance are sufficient. It highlights the robustness of the learning concept, in that it can learn even during a fall.

Figure 9.8 shows the relevant data that allows us to analyze the learning process in detail. The experiment lasted 30 seconds in total. A first push was applied to the robot after 14 seconds, and a second push was applied after 24 seconds. The pushes are marked with dotted vertical lines in the plots. The first and second plots show the trunk pitch and pitch rate respectively. The trunk pitch values show how the robot fell forward after the first push, but not after the second push. In the pitch rate signal, two arrows mark the first peak after each push, showing that both pushes were equally strong. The pitch rate signal is quite noisy due to the general shaking of the robot. After the second push, the pitch rate is negative at 25 seconds, when the robot steps forward and corrects the trunk pitch angle to an upright position. The third plot shows the gradient that was computed after each step. The gradient increases and the robot learns during the three steps after the first push, while the robot is falling forward, and

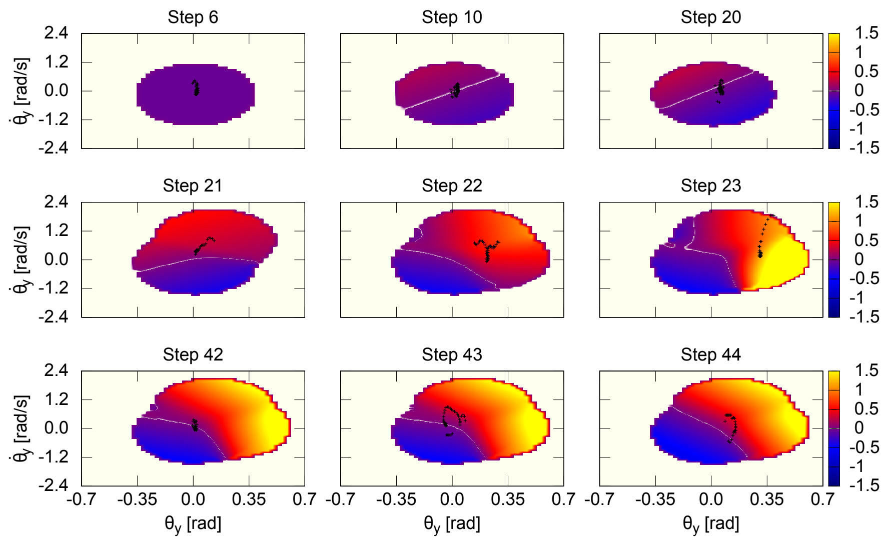


Figure 9.9: Evolution of the function approximator of the sagittal step size controller during the push recovery experiment. The black dots mark the data points that were used to update the function approximator at the end of the step.

larger and larger steps would have needed to be taken to regain balance. When the fall is detected at approximately 15.5 seconds, the gait controller switches to a fallen state, where walking and learning are suppressed. When an upright pose is detected, the controller automatically switches the walking and the learning back on. This is the case shortly before 20 seconds, where an undesired gradient can be seen to have been computed. Wrong gradients can be computed while the robot is standing, but is still in the process of manually being moved into the correct position. LWPR assigns more weight to new data and eventually forgets old data, and thus it makes sure that bad data does not degrade performance for an extended period of time. After the second push, the gradient stays near zero. The robot already learned the correct step size to cope with the push and no further adjustment is needed. The fourth plot shows the step size measured during the experiment. It is interesting to note that the robot responded with a smaller step size to the second push than to the first push, and yet the robot did not fall. The evident reason for this is that since the controller already learned from the first push, it was able to react earlier to the second push and could balance the robot with less effort, and more elegantly, with a single step. As can be inferred from the inverted pendulum dynamics, the sooner corrective action is taken, the smaller the correction needs to be.

The evolution of the function approximator during the learning experiment can be observed in Fig. 9.9. The black dots on top of the color maps of the function approximator show the data points that were used to update the controller after the preceding step. The first row consists of states of the function approximator before the first push. While the robot is walking in place, no significant change can be observed. We use a relatively large kernel size within the LWPR. The small cluster of data in the middle of the trunk pitch phase space is covered by a single linear kernel. The first push is followed by steps 21, 22, and 23 in the second row, during which the robot is falling. The LWPR algorithm places new kernels along the observed trajectory of the pitch angle state in the phase space and covers a large portion of it with non-zero leg swing amplitude activation values. In the last row, step 42 is the last step before the second push, where the robot is still stable. Step 43 is the capture step that absorbs the push, and finally, step 44 is a step of nearly zero size, during which the residual instability is depleted passively.

There are interesting insights to be gained from this experiment. The fact that the initialization with the analytic controller was not present shows that the learning technique for the sagittal step size is robust. It can be initialized with zero step sizes, and learn from data gathered during falling rather than during mostly stable walking. The implication of this is that the sagittal learning technique can be potentially used in constellations where a stable walking core is not present, and there is no other choice than to begin the learning process by tentative stepping and falling. The speed of learning from only a few steps is remarkable and unique among walking- and balance-related online learning attempts so far. Finally, the strength of the push recovery skills Copedo was able to obtain by learning is comparable, if not stronger, than the push recovery skills demonstrated by Dynaped with the analytic controller.

9.4 DISCUSSION

The load bearing concept of making the problem of learning to walk tractable for online machine learning is the decoupling of the whole-body walking motion from the concept of balance. When using an existing motion generator for stepping motions that can be controlled in its step size and timing, the complexity of the learning problem is reduced to the learning of a low-dimensional balance controller with reference tracking capabilities. Furthermore, the balance controller itself can be divided into isolated concepts of sagittal and lateral balance, and step timing. Our implementation of an analytic controller based on the dimensional decomposition paradigm is strong evidence that this technique can result in a significant improvement of balance with respect to open-loop walking. A set of simple, low-dimensional controllers are easier to learn than a complex one that attempts to tackle all control tasks at once.

To overcome the initial problem of a falling robot, we embedded our learning concept into an analytic gait generation framework. The analytic controller provides sufficient walking skills to begin the learning process without risking hardware damage, only an offset to the analytic controller needs to be learned in order to improve the walking performance of the robot.

We proposed to use a gradient-based update method to train function approximators that represent the controllers to be learned. To compute the gradient, we invest a model assumption and thereby limit the competence of the learning algorithm to inverted pendulum-like balancing tasks. Nevertheless, we gain a competitive learning performance that is able to balance a humanoid robot after a strong push. In the example of sagittal step size learning, a few steps were sufficient for the controller to produce convincing push recovery capabilities, even when the analytic controller is not present. To our knowledge, the speed of our learning concept as well as the achieved sagittal balance are among the best results accomplished on a bipedal robot to date.

The learned controller automatically accounts for latency and imprecise actuation as good as it can.

One of the most interesting aspects of using machine learning to solve the walking control task is that the latency and imprecise actuation are more or less taken into account automatically. In our analytic design of a bipedal walking controller we explicitely included a component that converts the commanded step size to CPG activation signals based on data collected from the robot that includes the imprecise execution of stepping motions. We also included a predictive filter to compensate for the latency. A learned controller automatically learns to account for these factors.

10 CONCLUSION

In this thesis we introduced a new approach to robust bipedal gait control with push recovery capabilities. The core concept of the gait controller is to use a Central Pattern Generator to generate open-loop stepping motions that can be controlled in terms of step size and timing. Using this control interface to abstract from high-dimensional whole-body control allows the implementation of a much simpler, low-dimensional balance controller that controls the Central Pattern Generator using Cartesian footstep coordinates and step timing. The balance controller is derived analytically with the help of the Linear Inverted Pendulum Model and can be computed efficiently in closed form. At the same time, the constraints of the low-dimensional balance model are not forced upon the robot and the natural dynamics of the bipedal walking motion can still be fully exploited. The execution of the whole-body walking motion is entirely up to the Central Pattern Generator. Its full complexity can be used to generate an energy efficient walking motion with stretched knees and a non-planar Center of Mass trajectory.

The main contributions of this thesis are a robust analytic gait controller and a fast online learning method for the learning of the sagittal step size.

In spite of the imprecision and latency caused by the compliant setting of the actuators, we were able to demonstrate robust and controllable omnidirectional walking with a real robot with potentially the strongest push recovery capabilities to date. We were able to accomplish this without using a precise dynamic model of the robot, without detecting foot contact, and without means of measuring or enforcing the model-suggested location of the Zero Moment Point. To our knowledge, the presented gait control framework is the first to take charge of step timing, and the first to be able to cope with oblique poses, thereby discarding the assumption that the feet of the robot have to remain flat on the floor. It is in fact one of the most surprising results that despite using only a planar model for balance, and a motion generator that has been designed to walk exclusively on a horizontal floor, the closed-loop gait controller is able to recover from situations where the robot has already tipped over the edge of a foot. We have substantiated this claim with systematic statistical experiments and video examples.

Furthermore, we proposed an online learning concept that can complement and improve the analytic controller. Our learning concept benefits from the same step size interface of the Central Pattern Generator that was used to simplify the task of designing the analytic footstep controller. The decomposition of the sagittal and the lateral dynamics simplifies the learning task further. We pragmatically identified initialization with a concept of balance as a key factor for successful learning in a real hardware environment, and introduced a gradient based learning technique that boosts the learning speed. We were able to demonstrate state of the art online learning performance of the sagittal step size with a real robot that, even when started without initialization, learns how to absorb a strong push from the experiences of a few failed steps during falling. The strength of the learned push recovery skills is comparable to, or better than what we were able to achieve with the analytic controller.

Apart from the general concept of the separation of motion and balance, we would like to highlight our method of Center of Mass state estimation with a kinematic model, our powerful predictive filter algorithm that removes noise and compensates latency, the mathematical framework for computing Zero Moment Point, step timing, and step location control variables, and the gradient based training of a function approximator as our in depth contributions in this thesis.

The proposed gait control concept is of manageable complexity and does not exhaust the available resources.

The architecture of the Capture Step Framework gives rise to potential beyond the concepts that have been researched so far. The manageable level of complexity of the Capture Step Framework leaves sufficient room to be extended with additional functionalities that seem directly within reach. For example, the controller could utilize the mass of the trunk and the arms as reaction mass and help the robot balance. It also seems obvious that the motion generator and the balance controller should be upgraded to take the inclination of the robot with respect to the floor explicitely into account, if not be able to generate stepping motions suitable for locomotion in non-level terrain. More complex learning algorithms could be added that operate in a higher dimensional input space and thereby take correlations into account, for example between the sagittal and the lateral directions, which are neglected by the simple learners proposed so far.

The balance control augmentation could be combined with a motion generator designed to be used in combination with series elastic actuators. Compliant, muscle-tendon-like actuation appears to be an integral part of the natural walking motion [Geyer et al., 2006] and is therefore a promising course for further investigation.

Separating the control of balance from the walking motion, and having a simple balance controller available that can potentially adapt to small changes of the walking motion, opens the perspective of attempting to optimize the walking motion directly on the hardware. Theoretical methods have been developed in simulation [Felis and Mombaur, 2013], but are not yet applicable to real robots. Starting with a robot that can already walk, and simplifying the optimization to not be concerned with balance could leverage an online optimization algorithm to find a motion pattern for the optimal use of the available actuators.

The Capture Step framework does not make prohibitive use of memory and computation time and leaves these resources available for computationally intensive processes. Footstep planning, for example, could be a well fitting extension to the short-sighted one step lookahead character of the capture step controller. Footstep planning offers the flexibility to traverse cluttered environments and rough terrain, and could utilize the available computational resources to plan several steps ahead. With the help of a simple physical model for balance, such as the one included in the analytic controller, a footstep planning algorithm could take the state of balance into account and plan multi-step recovery sequences while potentially avoiding prohibited foothold locations. The desired step size-based command interface of the Capture Step Framework seamlessly lends itself to be controlled by a footstep planning algorithm.

The Capture Step Framework can potentially support future research of elastic motion generation, online optimization of the walking motion, and balance aware footstep planning.

BIBLIOGRAPHY

Juan José Alcaraz-Jiménez, Marcell Missura, Humberto Martínez Barberá, and Sven Behnke. Lateral Disturbance Rejection for the Nao Robot. In *RoboCup*, pages 1–12, 2012.

Philipp Allgeuer and Sven Behnke. Fused angles for body orientation representation. In *Proceedings of 9th Workshop on Humanoid Soccer Robots of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, Madrid, Spain, 2014a.

Philipp Allgeuer and Sven Behnke. Robust sensor fusion for robot attitude estimation. In *Proceedings of 14th IEEE-RAS Int. Conference on Humanoid Robots (Humanoids)*, Madrid, Spain, 2014b.

S. O. Anderson, M. Wisse, C. G. Atkeson, J. K. Hodgins, G. J. Zeglin, and B. Moyer. Powered Bipeds Based on Passive Dynamic Principles. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2005.

Catherine E. Bauby and Arthur D. Kuo. Active Control of Lateral Balance in Human Walking. *Journal of Biomechanics*, 33(11):1433–1440, 2000.

Sven Behnke. Online trajectory generation for omnidirectional biped walking. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1597–1603, 2006.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman Hall, New York, 1984.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125.

J. Chestnutt, M. Lau, K.M. Cheung, J. Kuffner, J.K. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2005.

Steven H. Collins, Martijn Wisse, and Andy Ruina. A Three-Dimensional Passive-Dynamic Walking Robot with Two Legs and Knees. *International Journal of Robotics Research*, pages 607–615, 2001.

H. Diedam, D. Dimitrov, P.-B. Wieber, K. Mombaur, and M. Diehl. Online Walking Gait Generation with Adaptive Foot Positioning Through Linear Model Predictive Control. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

H. Dong, M. Zhao, and N. Zhang. High-Speed and Energy-Efficient Biped Locomotion Based on Virtual Slope Walking. *Autonomous Robots*, 30(2), 2011.

J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger. Bipedal Walking Control Based on Capture Point Dynamics. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

Martin Felis and Katja Mombaur. Modeling and optimization of human walking. In Katja Mombaur and Karsten Berns, editors, *Modeling, Simulation and Optimization of Bipedal Walking*, volume 18 of *Cognitive Systems Monographs*, pages 31–42. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-36367-2. URL http://dx.doi.org/10.1007/978-3-642-36368-9_3.

Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. Flexible Muscle-Based Locomotion for Bipedal Creatures. *ACM Transactions on Graphics*, 32(6), 2013.

Tao Geng, Bernd Porr, and Florentin Wörgötter. Fast biped walking with a sensor-driven neuronal controller and real-time online learning. *International Journal of Robotics Research*, 2006.

Hartmut Geyer and Hugh Herr. A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(3):263–273, 2010.

Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Compliant Leg Behaviour Explains Basic Dynamics of Walking and Running. *Proceedings of the Royal Society of London B: Biological Sciences*, 273 (1603):2861–2867, 2006. ISSN 0962-8452.

Colin Graf, Alexander Härtl, Thomas Röfer, and Tim Laue. A Robust Closed-Loop Gait for the Standard Platform League Humanoid. In *Workshop on Humanoid Soccer Robots*, 2009.

Inyong Ha, Yusuke Tamura, and Hajime Asama. Development of open platform humanoid robot darwin-op. *Advanced Robotics*, 27 (3):223–232, 2013.

T. Hemker, M. Stelzer, O. von Stryk, and H. Sakamoto. Efficient walking speed optimization of a humanoid robot. *International Journal of Robotics Research*, 28(2):303–314, 2009.

K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The Development of Honda Humanoid Robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1998.

Armin Hornung, Andrew Dornbush, Maxim Likhachev, and Maren Bennewitz. Anytime search-based footstep planning with suboptimality bounds. In *Proc. of the IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, Osaka, Japan, November 2012.

S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, and K. Yokoi. Biped Walking Pattern Generation by Using Preview Control of Zero-Moment Point. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2003.

S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi. Biped Walking Stabilization Based on Linear Inverted Pendulum Tracking. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3D linear inverted pendulum mode: a simple modeling for a bipedwalking pattern generation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2004.

Arthur D. Kuo. Stabilization of Lateral Motion in Passive Dynamic Walking. *The International Journal of Robotics Research*, 18(9):917–930, 1999.

Arthur D. Kuo. The Six Determinants of Gait and the Inverted Pendulum Analogy: A Dynamic Walking Perspective. *Human Movement Science*, 26(4):617 – 656, 2007. ISSN 0167-9457. European Workshop on Movement Science 2007 European Workshop on Movement Science 2007.

Arthur D. Kuo, J. Maxwell Donelan, and Andy Ruina. Energetic Consequences of Walking Like an Inverted Pendulum: Step-to-step Transitions. *Exercise and Sport Sciences Reviews*, 33(2):88–97, 2005. URL http://www.ncbi.nlm.nih.gov/pubmed/15821430.

J.-P. Laumond, G. Arechavaleta, T.-V.-A. Truong, H. Hicheur, Q.-C. Pham, and A. Berthoz. The Words of the Human Locomotion. In Makoto Kaneko and Yoshihiko Nakamura, editors, *Robotics Research*, volume 66 of *Springer Tracts in Advanced Robotics*, pages 35–47. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-14742-5.

Tad McGeer. Passive Dynamic Walking. *International Journal of Robotics Research*, 1990.

Marcell Missura and Sven Behnke. Lateral capture steps for bipedal walking. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2011.

Marcell Missura and Sven Behnke. Self-Stable Omnidirectional Walking with Compliant Joints. In *Workshop on Humanoid Soccer Robots*, Atlanta, USA, 2013a.

Marcell Missura and Sven Behnke. Omnidirectional Capture Steps for Bipedal Walking. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2013b.

Marcell Missura and Sven Behnke. Online Learning of Balanced Foot Placement for Bipedal Walking. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014a.

Marcell Missura and Sven Behnke. Balanced walking with capture steps. In *RoboCup 2014: Robot Soccer World Cup XVIII (to appear)*. Springer, 2014b.

Marcell Missura, C. Münstermann, P. Allgeuer, M. Schwarz, J. Pastrana, S. Schueller, M. Schreiber, and Sven Behnke. Learning to improve capture steps for disturbance rejection in humanoid soccer. In *RoboCup 2013: Robot Soccer World Cup XVII*, pages 56–67. Springer, 2014.

Jun Morimoto and Christopher G. Atkeson. Learning Biped Locomotion. In *IEEE Robotics and Automation Magazine*. IEEE, 2007.

Jun Morimoto and Christopher G. Atkeson. Nonparametric Representation of an Approximated Poincaré Map for Learning Biped Locomotion. In *Auton Robot*. Springer, 2009.

Jun Morimoto, Christopher G. Atkeson, Gen Endo, and Gordon Cheng. Improving Humanoid Locomotive Performance with Learnt Approximated Dynamics via guassian Process for Regression. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2007.

Mitsuharu Morisawa, Fumio Kanehiro, Kenji Kaneko, Nicolas Mansard, Joan Sola, Eiichi Yoshida, Kazuhito Yokoi, and Jean-Paul Laumond. Combining suppression of the disturbance and reactive stepping for recovering balance. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3150–3156, 2010.

Christian Ott, Christoph Baumgärtner, Johannes Mayr, Matthias Fuchs, Robert Burger, Dongheui Lee, Oliver Eiberger, Alin Albu-Schäffer, Markus Grebenstein, and Gerd Hirzinger. Development of a biped robot with torque controlled joints. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pages 167–173, 2010. ISBN 978-1-4244-8688-5.

I.-W. Park, J.-Y. Kim, J. Lee, and J.-H. Oh. Mechanical Design of Humanoid Robot Platform KHR-3 (KAIST Humanoid Robot 3: HUBO). In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2005.

Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual Model Control: An Intuitive Approach for Bipedal

Locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001.

Jerry E. Pratt, John Carff, Sergey V. Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pages 200–207. IEEE, 2006. ISBN 1-4244-0200-X.

Jerry E. Pratt, Twan Koolen, Tomas de Boer, John R. Rebula, Sebastien Cotton, John Carff, Matthew Johnson, and Peter D. Neuhaus. Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid. *The International Journal of Robotic Research*, 31(10):1117–1133, 2012.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

John Rebula, Fabian Canas, Jerry Pratt, and Ambarish Goswami. Learning Capture Points for Humanoid Push Recovery. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2007.

Max Schwarz and Sven Behnke. Compliant robot behavior using servo actuator models identified by iterative learning control. In *17th RoboCup International Symposium*, 2013.

Max Schwarz, Michael Schreiber, Sebatian Schueller, Marcell Missura, and Sven Behnke. Nimbro-op humanoid teensize open platform. In *Workshop on Humanoid Soccer Robots of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2012.

B. J. Stephens and C. G. Atkeson. Push Recovery by Stepping for Humanoid Robots with Force Controlled Joints. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

Russ Tedrake, Teresa Weirui Zhang, and H. Sebastian Seung. Learning to walk in 20 minutes. In *14th Yale Workshop on Adaptive and Learning Systems*, 2005.

J. Urata, K. Nishiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba. Online Decision of Foot Placement Using Singular LQ Preview Regulation. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2011.

Junichi Urata, Koichi Nishiwaki, Yuto Nakanishi, Kei Okada, Satoshi Kagami, and Masayuki Inaba. Online Walking Pattern Generation for Push Recovery and Minimum Delay to Commanded Change of Direction and Speed. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental Online Learning in High Dimensions. *Neural Comput.*, 17(12):2602–2634, December 2005. ISSN 0899-7667.

Jack M. Wang, Samuel R. Hamner, Scott L. Delp, Vladlen Koltun, and More Specifically. Optimizing Locomotion Controllers Using Biologically-Based Actuators and Objectives. *ACM Trans. Graph*, 2012.

Pierre-Brice Wieber. Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2006.

Martijn Wisse and J. Van Frankenhuyzen. Design and Construction of MIKE; a 2D Autonomous Biped Based on Passive Dynamic Walking. In *International Symposium of Adaptive Motion and Animals and Machines*, 2003.

Seung-Joon Yi, Byoung-Tak Zhang, Dennis Hong, and Daniel D. Lee. Online Learning of a Full Body Push Recovery Controller for Omnidirectional Walking. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2011.