Self-stable Omnidirectional Walking with Compliant Joints

Marcell Missura and Sven Behnke

Institute for Computer Science VI, Autonomous Intelligent Systems,
University of Bonn, Germany
missura@ais.uni-bonn.de

***Abstract*—Bipedal walking is one of the most essential skills in humanoid robot soccer. A stable and fast gait gives teams a winning edge when their robots are the first at the ball, maintain ball control with sure feet, and drive the ball decisively towards the opponent goal. The most successful teams in the Humanoid League are typically characterized by reliably walking robots.**

In this contribution, we describe the omnidirectional gait of team NimbRo, one of the most successful robot soccer teams in the history of RoboCup. The walk algorithm is open-loop and model-free. It is based on highly configurable, central-pattern-generated rhythmic motion signals and combines well with a compliant servo setting to achieve a relatively high level of self-stability. We discuss the advantages of this approach in comparison with methods of other successful teams and support our argumentation with experimental results.

I. Introduction

The ability to walk on two legs is the most distinguishing feature of humanoid robots. While legs are a desirable means of locomotion in terms of versatility and energy efficiency, the challenge of maintaining balance on rough terrain and in the presence of strong disturbances remains a difficult task for roboticists. In simplified, flat-floor environments, however, numerous teams of the Humanoid League are able to produce a relatively stable and dynamic walk for their bipedal robots and to provide for exciting games of robot soccer.

Team NimbRo 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 the years 2007 and 2008 and the TeenSize competitions from 2009 until 2013. One of the strengths of team NimbRo is a self-stable, omnidirectional gait that has been successfully adapted to a number of prototypes of varying sizes. Surprisingly, the gait is able to recover from disturbances, such as light pushes and stepping on small objects on the floor, even though it is completely open-loop.

It is certainly not our intention to claim that feedback is not necessary. To recover from large disturbances such as pushes and tripping, a quick and appropriate reaction is essential to maintain the balance of the biped and to return to a stable motion cycle. Recovery strategies include variation of step-timing, foot-placement, zero-moment-point control and the use of the upper body as a reaction mass, all of which require state feedback. But we argue that knowing where to step in the absence of disturbances is a trait of robustness. Using an open-loop stable core algorithm as starting point is beneficial for a number of reasons. Most importantly, it automatically yields a reference trajectory that can be described as a limit cycle of the system dynamics. Used as input for higher-level balance regulators, the natural dynamics of the physical system can be fully exploited and augmented with light-weight control strategies that try to return to a reference trajectory that the robot is comfortable with, rather than enforcing an unnatural gait that only works when feedback loops are active. The balance control loop can remain inactive and allow the robot to walk without control effort as long as corrective actions are not required. Furthermore, the open-loop motion trajectory can be used as fall-back in situations where sensor input may be too noisy, e.g., right after the support exchange, or in case of complete failure of sensory systems. Most of the successful walk algorithms use some sort of a reference trajectory [1], [2], [3]. However, they are not necessarily self-stable.

In the following, after reviewing related work, we detail our gait engine by describing the motion patterns that are combined to parametric omnidirectional stepping motions. We

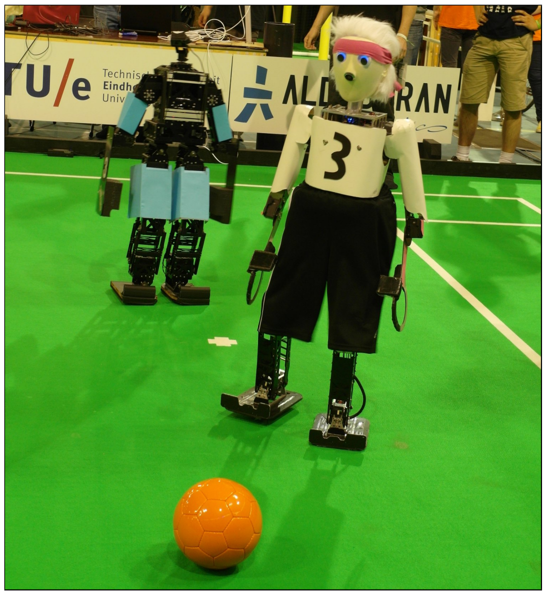
Fig. 1. NimbRo robot Copedo walking up to the ball.

provide an annotated set of configuration parameters as it is used for one of our robots. Then we show experimental results to demonstrate the open-loop stability that can be achieved with this approach. Finally, we discuss certain advantages of our approach in comparison to other popular algorithms.

II. Related Work

Zero moment point (ZMP) tracking with preview control [1] is the most popular scientific approach to bipedal walking to date. ASIMO [4], HRP-4C [5], and HUBO [6] are among the most prominent examples. A number of footsteps planned ahead in time are used as the reference input to define a desired ZMP trajectory. To generate a continuous center of mass (CoM) trajectory that minimizes the ZMP tracking error, optimization algorithms are generally used in a Model Predictive Control [7] setting. The CoM trajectory can then be used in combination with inverse kinematics or inverse dynamics to compute joint motions. These systems can walk reliably on flat ground and have the ability to cope with weak disturbances. However, the requirements of these algorithms, such as precise position tracking, accurate physical modeling, and high computational power, render them unsuitable for embedded systems combined with low-cost motors that are typically used by RoboCup teams.

In recent work, Urata et al. have presented an impressive foot-placement based controller on a real robot that is capable of recovering from strong pushes [8]. To speed up execution time, an LQ preview based algorithm is used to generate ZMP trajectories of a restricted form that can automatically be tracked by the CoM without the need for further optimization. To follow a commanded walking direction and speed, a minimum delay online modification of the ZMP trajectory was proposed [2]. While this controller successfully addressed the problems of push recovery and fast execution time, it still relies strongly on precise execution and an accurate full-body dynamics model for CoM trajectory tracking.

Englsberger et al. [3] presented a new approach to gait pattern generation based on capture point dynamics and showed how using the capture point as the reference input instead of the ZMP reduces the system equations to first order and eliminates the computationally expensive optimization of CoM trajectories. This approach is potentially suitable for robots competing on the soccer field. It has only been evaluated on the DLR biped [9] so far, but hopefully first implementations on other hardware will appear in the near future.

Focusing on the methods that are applied by leading teams of the Humanoid League, it is notable that the preferred algorithms are simple and light-weight in comparison to what is used in high-scale research projects in academic laboratory environments. The simplicity of an algorithm is an attractive feature when it comes to implementation on a real robot. It not only lowers the requirements on the expertise that is needed to successfully program and configure such an integral part of the robot operating software, but it is also highly correlated with computational efficiency. In the competitive and rapidly changing setting of RoboCup, the requirements on a walk algorithm shift towards the ability to adapt to walking surfaces, hardware modifications, and entirely new prototypes. The methods used by the RoboCup teams exhibit the necessary parameters to adapt the algorithms basically to any two-legged hardware. Algorithms that can optimize parameters in an automated manner, either online or during a training phase, have been experimented with [10], [11], [12], but are too time-consuming. Consequently, hand-tuning of parameters is often the method of choice to make robots walk. The gait tuning significantly influences robot performance during games.

Perhaps the most advanced closed-loop walk was presented for the Nao standard platform by Graf and Röfer [13], who proposed the online-adjustment of step parameters based on the solution of a system of linear pendulum equations. This is one of very few examples that takes the timing and the placement of foot steps into account. While a relatively weak open-loop core is present, the proposed feedback loop significantly increases the walking ability of the Nao robot to a level that has not been outperformed so far.

In the KidSize class, team DARwIn has been dominating the competitions in the recent years after the construction of the DARwIn-OP platform [14]. This capable hardware comes with a fast and reliable walk that is described in [15] along with an online learning algorithm that attempts to find stabilization parameters to cope with external disturbances. The core walking process has a strong similarity with ZMP based preview control. However, instead of the expensive CoM trajectory optimization that includes jerk minimization, the CoM trajectory is generated efficiently using simple linear inverted pendulum model equations. Swing foot trajectories are expressed as phase dependent trajectories in Cartesian space and are converted to joint motion using inverse kinematics.

Another remarkable gait generation technique has been proposed by the University of Tsinghua [16]. Inspired by the capability of passive dynamic walkers to walk down a shallow slope at the expense of minimal energy and no control effort at all, 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 for KidSize and AdultSize robots, as well as for a fast two-dimensional walker for scientific research.

In the closest related work [17], the motion patterns that are used by team NimbRo were first described. Since then, the algorithm has evolved. The patterns have been simplified and extended with new capabilities along with new configuration parameters that make the algorithm more flexible. It has been adopted to new hardware and servo compliance has been experimented with. In the recent years, the authors have investigated possibilities to augment the open-loop walk algorithm with closed-loop stabilizing strategies that are based on the already existing motion generator. The lateral balance

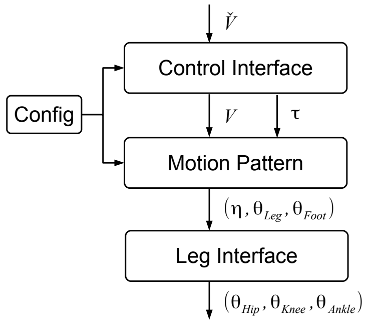
$\check{V}$
Control Interface
Config
V
τ
Motion Pattern
$(\eta, \theta_{Leg}, \theta_{Foot})$
Leg Interface
$(\theta_{Hip}, \theta_{Knee}, \theta_{Ankle})$

Fig. 2. Hierarchical layers of the NimbRo gait generator. The Control Interface receives a gait velocity target $\check{V}$ from a higher layer and translates it to a bounded and smoothed gait velocity vector V. The Motion Pattern layer generates phase τ dependent periodic motion signals on an abstract level that is translated by the Leg Interface to joint targets.

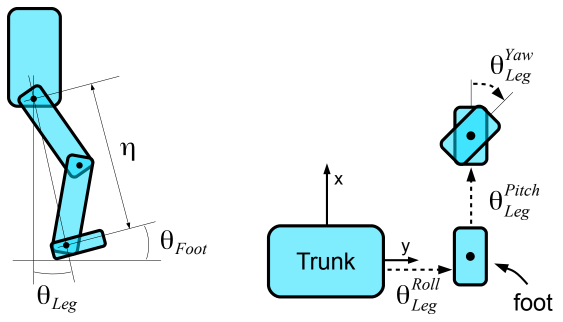

Fig. 3. The Leg Interface allows independent control of three abstract parameters: the leg extension η, the leg angle $\boldsymbol{\theta}_{Leg}$, and the foot angle $\boldsymbol{\theta}_{Foot}$ (left). The leg can be bent independently in roll, pitch, and yaw directions (right).

controller described in [18] was successfully implemented on a real robot and used in RoboCup games since 2011. An omnidirectional capture step controller [19] is currently being developed.

III. The NimbRo Gait Engine

The NimbRo gait generator algorithm can be represented with three logical layers, as illustrated in Figure 2. The highest instance is a Control Interface that bounds and smoothes a gait velocity target $\check{\boldsymbol{V}}$, the input into the gait engine from a higher layer. It produces a continuous gait velocity vector $\boldsymbol{V} = (V_x, V_y, V_\phi)$ with components in sagittal, lateral and rotational directions. The velocity vector is interpreted with respect to a right hand coordinate frame with the convention that the x-axis points in forward direction. The z-axis, the axis of the rotational gait component, points upwards. The Control Interface also maintains a motion phase τ and passes it on to the Motion Pattern layer. The Motion Pattern layer generates periodic motion signals that are modulated with the gait velocity input to produce omnidirectional stepping motions that result in the desired walking speed. The motion signals operate on an abstract level constituted by intuitive leg control parameters, such as the angle of the leg relative to the trunk, and the extension of the leg. These parameters are then translated to joint targets by the Leg Interface abstraction layer. A set of configuration variables is an integral part of the algorithm to allow easy hardware adaptation. The three layers are best explained in a bottom up order in more detail.

A. Leg Interface

The presented gait generation algorithm is entirely based on a low-level motion abstraction layer that we dubbed as the Leg Interface

$$L(\eta, \boldsymbol{\theta}_{Leg}, \boldsymbol{\theta}_{Foot}) = (\boldsymbol{\theta}_{Hip}, \theta_{Knee}, \boldsymbol{\theta}_{Ankle}). \tag{1}$$

The Leg Interface allows intuitive control of a leg with three input parameters. The leg extension η defines the distance between the foot and the trunk, the leg angle $\boldsymbol{\theta}_{Leg} = (\theta^{Roll}_{Leg}, \theta^{Pitch}_{Leg}, \theta^{Yaw}_{Leg})$ defines the angle of the leg with respect to the trunk, and the foot angle $\boldsymbol{\theta}_{Foot} = (\theta^{Roll}_{Foot}, \theta^{Pitch}_{Foot})$ defines the inclination of the foot with respect to the transversal plane. The leg can be bent in roll, pitch, and yaw directions, while the foot can be bent in roll and pitch directions. The meaning of these parameters is illustrated in Figure 3. The Leg Interface encapsulates the calculation of coordinated joint angles and computes an output for the hip, knee, and ankle joints according to the formulas

$$\begin{aligned}
\lambda &= \arccos(1-\eta), & (2)\\
\theta^{Yaw}_{Hip} &= \theta^{Yaw}_{Leg}, & (3)\\
\theta^{Roll}_{Hip} &= \theta'^{\,Roll}_{Leg}, & (4)\\
\theta^{Pitch}_{Hip} &= \theta'^{\,Pitch}_{Leg} - \lambda, & (5)\\
\theta_{Knee} &= 2\lambda, & (6)\\
\theta^{Pitch}_{Ankle} &= \theta^{Pitch}_{Foot} - \theta'^{\,Pitch}_{Leg} - \lambda, & (7)\\
\theta^{Roll}_{Ankle} &= \theta^{Roll}_{Foot} - \theta'^{\,Roll}_{Leg}, & (8)
\end{aligned}$$

with

$$\begin{bmatrix} \theta'^{\,Pitch}_{Leg} \\ \theta'^{\,Roll}_{Leg} \end{bmatrix} = R(-\theta^{Yaw}_{Leg}) \begin{bmatrix} \theta^{Pitch}_{Leg} \\ \theta^{Roll}_{Leg} \end{bmatrix}, \tag{9}$$

where R is a two-dimensional rotation matrix.

An important feature of the Leg Interface is the independence of the input parameters. In particular this means that rotations in yaw direction are interpreted with respect to the center of the foot instead of with respect to the hip, where in the latter case a rotation would have an influence on the sagittal and lateral position of the foot and thus interfere with the effect of the θ^{Pitch}_{Hip} and θ^{Roll}_{Hip} parameters. This feature is implemented by equation (9). The parameter independence makes it easier to control foot configurations on a higher layer.

The leg extension parameter $\eta \in [0, 1]$ is expected to be in a bounded range, where a leg extension of 0 is interpreted as a fully extended leg and the leg extension of 1 is interpreted as a fully retracted leg. The reason why the parameter was assigned this way is a convention that values of 0, no matter if Leg Interface parameters or joint angles, result in a safe robot pose where the legs are fully extended parallel to the trunk and to each other, and the feet are orthogonal to the legs. 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 single joint basis.

The kinematic order of the joints is relevant for the computation of the joint angle output. The formulas (3) to (8) assume that the joint sequence starting from the hip is: hip yaw, hip roll, hip pitch, knee pitch, foot pitch, and foot roll. Additionally, the Leg Interface assumes that the thigh and the shank are of equal lengths. In case of a different kinematic configuration, the Leg Interface formulas need to be adjusted accordingly.

B. Motion Pattern

The NimbRo gait is essentially a combination of rhythmic activation signals that encode periodic leg-lifting and leg-swinging motions. The output of the pattern generator is a set of Leg Interface parameters that allow us to conveniently design leg angle and leg extension trajectories instead of having to think about coordinated motions on the single joint level. The motion pattern can be subdivided into isolated motion primitives, which will be presented in the following. We denote motion primitives with the letter P and configuration variables with the letter C.

1) Halt Position: The halt position is a static, i.e., not phase dependent offset from zero. It is a pose the robot is standing in when the robot is not walking. 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 P_{Halt} is a collection of configuration parameters

$$\begin{aligned} P^{\eta}_{Halt} &= C_1, \\ P^{LegRoll}_{Halt} &= \sigma C_2, \\ P^{LegPitch}_{Halt} &= C_3, \\ P^{FootRoll}_{Halt} &= C_4, \\ P^{FootPitch}_{Halt} &= C_5, \end{aligned}$$

where $\sigma \in \{-1, 1\}$ denotes the leg sign (left or right).

2) Leg Lifting: The alternating shortening of a leg induces a lateral oscillation of the body mass and frees one leg at a time from its support duty. The leg extension is activated with a sinusoidal function

$$P_{LegLift} = \begin{cases} \sin(\tau)\,(C_6 + C_7\,\max(|V_x|, |V_y|)), & \tau \le 0 \\ \sin(\tau)\,(C_8 + C_9\,\max(|V_x|, |V_y|)), & \text{else} \end{cases}, \tag{10}$$

that takes the motion phase $\tau \in [-\pi, \pi)$ as an argument. The motion phase of the left leg is shifted by π with respect to the right leg, so that the same motion pattern can be used for both legs at the same time.

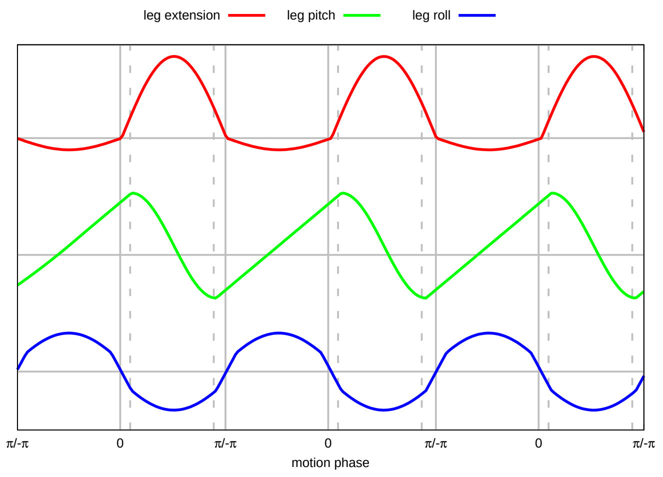

Fig. 4. The main ingredients of the gait motion are rhythmical leg lifting (top), leg swing motion (center), and a lateral hip swing (bottom). Only the patterns for the right leg are shown. The solid vertical lines indicate the expected times of support exchange. The dashed vertical lines indicate the swing start and swing end timings.

The leg lifting motion is shown in Figure 4 on the top. Notably, the leg lifting primitive makes a distinction between a support phase ($\tau \le 0$), when the foot is on the ground, and a swing phase ($\tau > 0$), when the foot is in the air and can be swung. The configuration variables C_6 and C_7 describe a constant push amplitude and a gait velocity dependent push amplitude during the support phase. The step height amplitudes during the swing phase are determined by C_8 and C_9, respectively. Support exchange is expected to occur at motion phase $\tau = 0$ and $\tau = \pm\pi$. 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 be no greater than the leg extension in the halt position C_1, otherwise the leg would be overstretched. The velocity dependent increase of the step amplitude is crucial to avoid ground contact when a larger step size is used and allows for calm steps with a low height when the robot is walking slowly or on the spot.

3) Leg Swing: To induce a walking motion in any direction, a leg swing pattern

$$\gamma = \begin{cases} \cos\left(\frac{\tau - C_{\tau_0}}{C_{\tau_1} - C_{\tau_0}}\pi\right), & C_{\tau_0} \le \tau < C_{\tau_1} \\ \frac{2(\tau - C_{\tau_1})}{2\pi - C_{\tau_1} + C_{\tau_0}} - 1, & C_{\tau_1} \le \tau < \pi \\ \frac{2(\tau + 2\pi - C_{\tau_1})}{2\pi - C_{\tau_1} + C_{\tau_0}} - 1, & -\pi \le \tau < C_{\tau_0} \end{cases}, \tag{11}$$

$$P^{Pitch}_{LegSwing} = \begin{cases} \gamma V_x C_{10}, & V_x \ge 0 \\ \gamma V_x C_{11}, & V_x < 0 \end{cases}, \tag{12}$$

$$P^{Roll}_{LegSwing} = -\gamma V_y C_{12} - \sigma \max(|V_y| C_{13}, |V_\phi| C_{14}), \tag{13}$$

$$P^{Yaw}_{LegSwing} = \gamma\, V_\phi\, C_{15} - \sigma |V_\phi| C_{16} \tag{14}$$

is used. As shown in Figure 4 in the center, the leg is swung forward with a sinusoidal motion and pushed back with a linear motion in the support phase. The motivation why the support motion has been designed this way is that if the legs were the spokes of a wheel that is traveling with a constant speed, the angular velocity of the spokes would also be constant. The free 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 motion phase. Swing phase configuration parameters C_{τ_0} and C_{τ_1} are used to delay the start of the swing motion and to rush the touch down of the leg before the support exchange at motion phase $\tau = \pm\pi$. We found that these parameters are a good way to eliminate shuffling and to implicitly create a short double support phase, that we do not take into account otherwise. Again, we delay the motion phase of the left leg by π with respect to the right leg and use the same motion pattern to drive both legs at the same time.

The equations to produce the leg swing motion differ in the three directions. In sagittal direction (12), the legs are encouraged to swing fully from front to back. Separate swing amplitudes for walking forward and backward are configured using the step size parameters C_{10f} and C_{11}. In lateral direction (13), however, the legs would collide. Therefore, leg roll angle offsets C_{13} and C_{14} are added proportionally to the lateral and rotational gait velocities V_y and V_ϕ, causing the legs to spread out when walking in lateral direction and when the robot is turning. In rotational direction (14), a velocity dependent yaw angle offset can be configured using the parameter C_{16}. An annotated list of all configuration parameters is given in Table 1.

4) Lateral Hip Swing: The lateral hip swing sways the pelvis left and right during walking and helps to transfer the CoM from leg to leg at the right time. The hip swing is designed as two opposing sinusoid motions, one for the hip swing to the left and one for the hip swing to the right. These sinusoids are also not perfectly embedded into the motion phase. Two explicit motion phases τ_l and τ_r

$$\tau_l = \begin{cases} \tau - C_{\tau_1} + 2\pi, & \tau < C_{\tau_0} \\ \tau - C_{\tau_1}, & \tau > C_{\tau_1} \\ 0, & \text{else} \end{cases}, \tag{15}$$

$$\tau_r = \begin{cases} \tau - C_{\tau_1} + 3\pi, & \tau + \pi < C_{\tau_0} \\ \tau - C_{\tau_1} + \pi, & \tau + \pi > C_{\tau_1} \\ 0, & \text{else} \end{cases}, \tag{16}$$

are derived from the motion phase τ for the left swing (τ_l) and the right swing (τ_r) using the swing start C_{τ_0} and swing stop C_{τ_1} timing parameters to determine the start and end points of the sinus waves in the motion phase. 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 two sinusoids are summed up to create the final hip swing motion primitive

$$P_{HipSwing} = C_{17}(\sin(\tau_l \frac{\pi}{\delta}) - \sin(\tau_r \frac{\pi}{\delta})), \tag{17}$$

using $\delta = C_{\tau_0} - C_{\tau_1} + 2\pi$. The two swing patterns overlap in the double support phase and their sum creates a smooth transition between the left and the right swing phases, as illustrated in Figure 4. In the case of the hip swing motion primitive, the motion phase is not delayed for the left leg. Both legs execute the same pattern at the same time.

5) Leaning: The leaning motion primitive

$$P_{Lean}^{Pitch} = \begin{cases} V_x C_{18}, & V_x \geq 0 \\ V_x C_{19}, & V_x < 0 \end{cases}, \tag{18}$$

$$P_{Lean}^{Roll} = -V_\phi |V_x| C_{20}, \tag{19}$$

leans the robot slightly in the walking direction by adding an offset to the roll and pitch angles of the leg proportionally to the walking velocity. It does not depend on the motion phase. We differentiate between forward and backward walking and use separate parameters to calibrate the lean offset. The lean in lateral (roll) direction is determined by the combination of the sagittal and the rotational walking velocity. Practically speaking, the robot "leans into curves". In earlier work we performed an experiment with the NimbRo-OP [20] that demonstrates the benefit of the leaning primitive for the balance of the robot.

The final motion pattern is a sum of all motion primitives

$$\begin{aligned} \eta &= P_{Halt}^{\eta} + P_{LegLift} \\ \theta_{Leg}^{Roll} &= P_{Halt}^{LegRoll} + P_{HipSwing} + P_{LegSwing}^{Roll} + P_{Lean}^{Roll} \\ \theta_{Leg}^{Pitch} &= P_{Halt}^{LegPitch} + P_{LegSwing}^{Pitch} + P_{Lean}^{Pitch} \\ \theta_{Leg}^{Yaw} &= P_{LegSwing}^{Yaw} \\ \theta_{Foot}^{Roll} &= P_{Halt}^{FootRoll} \\ \theta_{Foot}^{Pitch} &= P_{Halt}^{FootPitch}. \end{aligned}$$

For the sake of brevity, we have neglected the description of the arm motion. However, the implementation of the arm motion within this pattern generator is straight forward. Similar to the Leg Interface, an Arm Interface would provide an abstract actuator space in which the same configurable swing motion primitive can be used, as it was already used to swing the legs.

C. Control Interface

The highest instance of the gait generator is a Control Interface layer that accepts the input of a desired walking velocity expressed in SI units. Using a simple linear mapping, we translate the input to a gait velocity vector $\check{\boldsymbol{V}} = (\check{V}_x, \check{V}_y, \check{V}_\phi) \in [-1, 1]^3$, where a value of 1 represents the highest achievable velocity. To implement omnidirectional walking, the three directional components can be arbitrarily combined as long as the velocity vector is contained within a convex region that is defined by a p-norm

$$\hat{\boldsymbol{V}} = \begin{cases} \frac{\check{\boldsymbol{V}}}{\|\check{\boldsymbol{V}}\|_{C_{21}}}, & \|\check{\boldsymbol{V}}\|_{C_{21}} > 1 \\ \check{\boldsymbol{V}}, & \text{else} \end{cases}. \tag{20}$$

The configurable shape of the applied norm enforces a restriction on the allowed velocity component combination and also bounds the gait velocity to remain inside the $[-1, 1]^3$ space, which may not yet be the case after the linear mapping of the SI input.

Essentially, the gait velocity vector determines the leg swing amplitudes in roll, pitch and yaw directions. Through the modulation of the motion pattern amplitudes in (10), (12), (13), (14), (18), and (19), the gait velocity has a direct effect on the motor targets that are sent to the robot. Therefore it is crucial that the Control Interface presents a gait velocity to the Motion Pattern layer that remains continuous at all times, otherwise the continuity of joint motions would be disrupted. The Control Interface maintains an internal state V of the gait velocity that is moved towards the bounded gait input $\hat{\boldsymbol{V}}$ in small increments within configurable bounds

$$V_x = V_x + \max(-C_{22}, \min(\hat{V}_x - V_x, C_{22})), \quad (21)$$
$$V_y = V_y + \max(-C_{23}, \min(\hat{V}_y - V_y, C_{23})), \quad (22)$$
$$V_\phi = V_\phi + \max(-C_{24}, \min(\hat{V}_\phi - V_\phi, C_{24})). \quad (23)$$

Through the allocation of the gait velocity bounding and smoothing in the Control Interface, higher control layers are relieved from the responsibility of protecting the robot from discontinuous motion signals.

Aside from the gait velocity, the motion phase τ is also maintained inside the Control Interface

$$\tau = \tau + C_{25} + |V_x| C_{26} + |V_y| C_{27}. \quad (24)$$

The parameters C_{26} and C_{27} allow velocity dependent step frequency modulation. Typically, the step frequency is slightly increased with sagittal velocity and slightly decreased with lateral velocity. Whenever the motion phase exceeds the upper bound of π, it needs to be wrapped

$$\tau = \begin{cases} \tau - 2\pi, & \tau > \pi \\ \tau, & \text{else} \end{cases}. \quad (25)$$

This cannot be done smoothly. Thus, all motion phase dependent motion primitives have to be designed carefully in order to produce a continuous output, even when the motion phase is reset.

TABLE I
AN ANNOTATED SET OF CONFIGURATION PARAMETERS THAT ARE USED BY THE GAIT GENERATION ALGORITHM. THE PROVIDED VALUES ARE A SET OF PARAMETERS THAT WE USE FOR OUR BIPEDAL ROBOT DYNAPED.

Variable	Value	Denotation
C_1	0.02	Halt Position Leg Extension
C_2	0.1	Halt Position Leg Roll Angle
C_3	0.02	Halt Position Leg Pitch Angle
C_4	0.03	Halt Position Foot Roll Angle
C_5	0	Halt Position Foot Pitch Angle
C_6	0.02	Constant Ground Push
C_7	0	Proportional Ground Push
C_8	0.3	Constant Step Height
C_9	0.12	Proportional Step Height
C_{τ_0}	0	Swing Start Timing
C_{τ_1}	2.3876	Swing Stop Timing
C_{10}	0.17	Sagittal Swing Amplitude Fwd
C_{11}	0.12	Sagittal Swing Amplitude Bwd
C_{12}	0.1	Lateral Swing Amplitude
C_{13}	0.05	Lateral Swing Amplitude Offset
C_{14}	0.015	Turning Lateral Swing Amplitude Offset
C_{15}	0.2	Rotational Swing Amplitude
C_{16}	0.05	Rotational Swing Amplitude Offset
C_{17}	0.035	Lateral Hip Swing Amplitude
C_{18}	0	Forward Lean
C_{19}	0	Backward Lean
C_{20}	-0.07	Forward and Turning Lean
C_{21}	3.5	Gait Velocity Limiting Norm p
C_{22}	0.0085	Sagittal Acceleration
C_{23}	0.01	Lateral Acceleration
C_{24}	0.009	Rotational Acceleration
C_{25}	0.09	Constant Step Frequency
C_{26}	0.008	Sagittal Proportional Step Frequency
C_{27}	0	Lateral Proportional Step Frequency

IV. Experimental Results

The accompanying video [21] shows a number of different humanoid robots that the walk algorithm described in this work has been implemented on. All of these robots could walk on a flat floor and demonstrated outstanding performance in RoboCup soccer games. In push experiments that were performed in the video, the biped was able to absorb impacts that were strong enough to create a visible disturbance during a completely open-loop walk. The robot was also able to step on a board with a thickness of 2 cm. Furthermore, the video shows some of the effects of compliant control. The robot automatically yields to pressure and dampens undesired swinging. While it is difficult to isolate the effect on stability, the ability to absorb shocks and to dampen swinging can be reasonably assumed to have a positive influence. Other effects we observed since we introduced compliant control are longer operation times before overheating and a reduction of damage to transmission gears.

Elastic actuation is achieved by using the compliance slope feature offered by the Dynamixel servos. The compliance slope has an influence on the gain of the PD-controllers that drives the servos to commanded positions. The higher the slope, the higher the gain when the current position deviates from the commanded position. We use a compliance slope of 32 according to the Dynamixel communication protocol for all motors in the leg. We have only been able to achieve this compliance level in combination with our prototypes that were

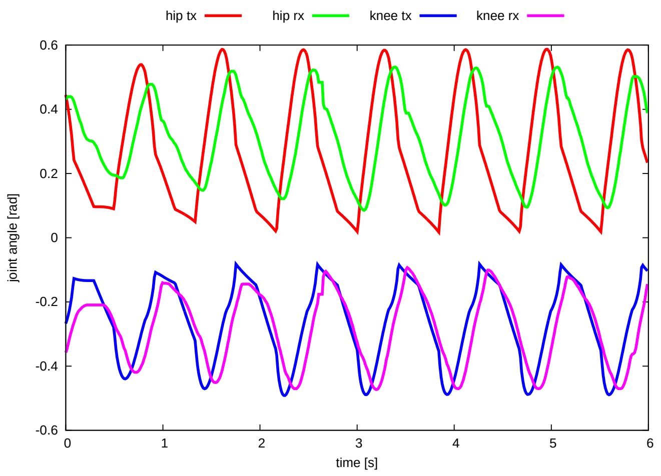

Fig. 5. Time series of the commanded (tx) positions and measured (rx) positions of the right hip pitch and knee joints during forward walking.

built using a parallel kinematic mechanical structure. Other robots had to be driven with a harder motor configuration. While the high motor compliance results in smooth and stable motions, it is important to be aware of the fact that the position tracking error of the motors increases. Figure 5 shows the time series of commanded and measured motor positions for the hip pitch and knee servos in a situation when the robot was walking forward with full velocity. The position tracking error as well as the actuation latency are evident. This is not an issue for the gait generator that was presented in this paper, because it is easy to reconfigure the commanded motion signals in a way that the desired output of the motors is achieved. However, in a recent publication [22] we have addressed this issue and proposed a possible solution to compliant actuation with precise position tracking.

V. Conclusion and Discussion

We have presented an algorithm that generates a compliant, open-loop walking motion for bipedal robots. The presented method has a number of advantages compared to other commonly accepted approaches. Our method is model free. Masses, inertia, and sizes of body parts do not need to be modeled. It combines well with elastic position control and it does not have an issue with position tracking errors. It moves the robot using an abstract forward kinematic interface and can produce a natural looking and energy-efficient robotic walk with stretched knees. Inverse kinematics and linear inverted pendulum based motion control algorithms typically restrict the motion of the center of mass to a plane and thus produce unnatural looking walking motions with bent knees that expose the leg motors to unnecessary strain. Using inverse kinematics to follow trajectories defined in Cartesian space also has the disadvantage that swing leg motions have to be designed in a way that they remain inside the kinematically feasible area of the legs. When using forward kinematics to express motions, this is always guaranteed.

We have demonstrated in an experiment that the walk is self-stable. It is able to absorb small disturbances without a feedback loop and to return to a stable cycle. This feature makes this central pattern generated motion controller an attractive building block in a hierarchical push recovery control approaches [18], [19].

In future work, we will continue to investigate feedback control strategies that are using this motion generator as a parameterized step controller to achieve a controllable, push robust gait.

VI. Acknowledgement

This work has been supported partially by grant BE 2556/6-1 of German Research Foundation (DFG).

References

[1] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, and Kazuhito Yokoi. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1620–1626, 2003.

[2] 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)*, pages 3411–3416, 2012.

[3] Johannes Englsberger, Christian Ott, Máximo A. Roa, Alin Albu-Schäffer, and Gerhard Hirzinger. Bipedal walking control based on capture point dynamics. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4420–4427, 2011.

[4] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of honda humanoid robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1998.

[5] Shuuji Kajita, Mitsuharu Morisawa, Kanako Miura, Shinichiro Nakaoka, Kensuke Harada, Kenji Kaneko, Fumio Kanehiro, and Kazuhito Yokoi. Biped walking stabilization based on linear inverted pendulum tracking. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4489–4496, 2010.

[6] Ill-Woo Park, Jung-Yup Kim, Jungho Lee, and Jun-Ho Oh. Mechanical design of humanoid robot platform khr-3 (kaist humanoid robot 3: Hubo). In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pages 321–326, 2005.

[7] 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)*, pages 137–142, 2006.

[8] Junichi Urata, Koichi Nishiwaki, Yuto Nakanishi, Kei Okada, Satoshi Kagami, and Masayuki Inaba. Online decision of foot placement using singular lq preview regulation. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pages 13–18, 2011.

[9] Christian Ott, Christoph Baumgrtner, Johannes Mayr, Matthias Fuchs, Robert Burger, Dongheui Lee, Oliver Eiberger, Alin Albu-Schffer, 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. IEEE, 2010.

[10] Felix Faber and Sven Behnke. Stochastic optimization of bipedal walking using gyro feedback and phase resetting. In *Humanoid Robots, 7th IEEE-RAS International Conference on*, pages 203–209, 2007.

[11] Cord Niehaus, Thomas Röfer, and Tim Laue. Gait optimization on a humanoid robot using particle swarm optimization. In *2nd Workshop on Humanoid Soccer Robots at Humanoids Conference*, Pittsburgh, 2007.

[12] Andrea Cherubini, Francesca Giannone, Luca Iocchi, M Lombardo, and Giuseppe Oriolo. Policy gradient learning for a humanoid soccer robot. *Robotics and Autonomous Systems*, 57(8):808–818, 2009.

[13] Colin Graf, Alexander Härtl, Thomas Röfer, and Tim Laue. A robust closed-loop gait for the standard platform league humanoid. In *4th Workshop on Humanoid Soccer Robots at Humanoids Conference*, 2009.

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

[15] 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)*, pages 1–6, 2011.

[16] Hao Dong, Mingguo Zhao, and Naiyao Zhang. High-speed and energy-efficient biped locomotion based on virtual slope walking. *Autonomous Robots*, 30(2):199–216, 2011.

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

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

[19] Marcell Missura and Sven Behnke. Omnidirectional capture steps for bipedal walking. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2013.

[20] 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)*, 20012.

[21] Marcell Missura and Sven Behnke. Demonstration of an open-loop, central-pattern-generated walk. http://www.ais.uni-bonn.de/movies/NimbRoGait.wmv.

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