Noname manuscript No.
(will be inserted by the editor)

Versatile and Robust Bipedal Walking in Unknown Environments

Real-Time Collision Avoidance and Disturbance Rejection

Arne-Christoph Hildebrandt · Robert Wittmann · Felix Sygulla · Daniel Wahrmann · Daniel Rixen · Thomas Buschmann

Received: date / Accepted: date

Abstract Autonomous navigation in complex environments featuring obstacles, varying ground compositions, and external disturbances requires real-time motion generation and stabilization simultaneously. In this paper, we present and evaluate a strategy for rejection of external disturbances and real-time motion generation in the presence of obstacles and non-flat ground. We propose different solutions for combining the associated algorithms and analyze them in simulations. The promising method is validated in experiments with our robot LOLA. We found a hierarchical approach to be effective for solving these complex motion generation problems, because it allows us to decompose the problem into sub-problems that can be tackled separately at different levels. This makes the approach suitable for real-time applications and robust against perturbations and errors. Our results show that real-time motion planning and disturbance rejection can be combined to improve the autonomy of legged robots.

1 Introduction

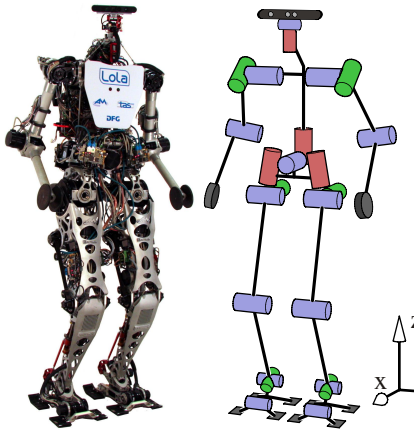
Although companies are starting to show increasing interest in the topic, bipedal robots are still the subject of fundamental research. Legged robots will only be used in real applications if they are able to reliably navigate through complex scenarios. We expect complex scenarios to raise multiple issues for robotic systems: (1) motion generation problems in complex environments have to be solved autonomously; (2) robots have to be able to react quickly to dynamically changing environments or user input; (3) robots have to be reliable and robust, even when subjected to unknown disturbances or modeling errors. Recent research proposes sophisticated solutions for each of these issues: during the DARPA challenge, legged robots navigated through static environments via teleoperation (Fallon et al, 2015; Stumpf et al, 2014). Changing user input and dynamically changing environments are furthermore taken into account in real-time methods presented in Chestnutt et al (2009); Buschmann (2010). The authors of Nishiwaki and Kagami (2009b); Takenaka et al (2009) presented methods to reject large disturbances such as pushes or unmodeled terrain while walking. However, all three solutions need to perform simultaneously when navigating in complex scenarios. We call this *versatile and robust walking*.

- By **versatility**, we refer to strategies enabling the robot to efficiently use its full physical capabilities in various situations. This is important when stepping over or onto complex, previously unknown obstacles or when walking with large step lengths.
- By **robustness**, we mean the robot's ability to recover from severe disturbances resulting from external forces or errors in the environment model. Only large modifications of the system's originally planned motion can achieve such robustness.

In this article, we present our hierarchical approach to make humanoid walking more *versatile* and *robust*. Furthermore, we present and analyze ways to combine these methods, which will enable autonomous walking in complex scenarios.

We first consider *robust* and *versatile* walking strategies separately, and then, we present their com-

Arne-Christoph Hildebrandt · Robert Wittmann
Technical University of Munich,
85747 Garching, GER
Phone: +49 89289 15208
E-mail: arne.hildebrandt@tum.de
E-mail: robert.wittmann@tum.de



Joint	DoF
Head	2
Shoulder	2
Elbow	1
Pelvis	2
Hip	3
Knee	1
Ankle	2
Toe	1
Total	**24**

Fig. 1 Photo and kinematic structure of the humanoid robot *Lola*. Joint distribution is shown on the right side.

bination. In Section 2, we provide an overview of current related research projects. Section 3 presents a system overview of the experimental platform used in this work, the robot LOLA, and its overall software framework for *versatile* and *robust* walking. Our main contribution is divided into three sections: first, we present our methods for real-time motion generation and disturbance rejection in Section 4 and in Section 5, respectively. Then, in Section 6, we present methods to tightly integrate both. Different methods are further analyzed in simulation and the most promising one is validated successfully in the experiments conducted. Experimental results are presented in Section 7. Finally, Section 8 is devoted to a conclusion as well as comments on limitations and future work.

2 Literature

Methods for *versatile* walking and *robust* walking are mostly handled separately in the state-of-the-art. These methods are thus reviewed first, followed by published work that combines both approaches.

2.1 Versatile Walking

To achieve autonomous navigation in real-time, the humanoid robot's motion generation is typically divided hierarchically to reduce computational cost. Footholds are determined first, to be able to take into account cluttered environments. The multi-body system's motion trajectories are planned based on these. Okada et al (2001) presented early work on autonomous walking. Based on a 2.5D approximation of the environment, the robot *H7* was able to navigate collision-free in the presence of obstacles. Sabe et al (2004); Michel and Chestnutt (2007); Gutmann et al (2008) improved this concept. In 2004, Pfeiffer and Schmidt presented initial experiments with the robot *Johnnie*, using an on-board vision system to autonomously step over obstacles or onto stairs (Lohmeier et al, 2004). This work produced impressive results, but their strong assumptions about the environment limited the concept they presented. Chestnutt et al. presented more generally applicable methods. They set up a laser scanner to construct an accurate height map of the environment ((Chestnutt et al, 2006, 2009)) and an external motion capture system to localize the robot. Using the height map, an A*–search was applied to find valid footholds. At the *DARPA Robotics Challenge* Trails, Fallon et al (2015) developed an interesting method differing from the previous graph-search-based approaches. Instead of representing the environment based on a 2.5D map, the vision system determines convex areas into which the robot is able to step. Starting with a fixed number of steps, the step planning problem can be solved using mixed integer optimization. Buschmann et al (2010) presented a different concept. Instead of using a global map of the environment, they proposed analyzing a set of 2D trajectories for feasibility. Using this method, the robot *Lola* was able to navigate autonomously through an unknown and changing environment. The presented implementation limited the robot to 2D paths and did not exploit the robot's ability to step over obstacles. Recently, Karkowski and Bennewitz (2016) proposed a similar approach. They first plan a global 2D path using an A*–Search. Then sub-goals connected by line segments are generated from the 2D path. The line segments are subsequently used by a local step planner for geometrically generating the foothold positions. However, the robot's kinematic capacities, such as stepping over obstacles, are not fully exploited, nor is the solution's optimality explicitly considered. The methods mentioned so far all concentrate on finding valid step sequences. Collision avoidance is taken into account for motion generation only via heuristics. Perrin et al (2012) proposed dividing the stepping movements into static half-steps. These half-steps are analyzed for collisions using collision geometries calculated offline. For reaching approximately dynamic movements, the half-steps are smoothed during a subsequent step. The whole planning process needs 2.5 s, which entails high latencies for fast, dynamic walking. In contrast to Perrin et al (2012), Maier et al (2013) combine perception, path planning and collision-free motion generation. They use an inverse height map of the robot *Nao* for step suggestions in an A*-search based step planner to check for collisions in discrete configurations. Complementary to the presented approaches, another body

of literature focuses directly on the stepping motion. Yisheng Guan et al (2005); Guan et al (2006) present a method for collision-free, quasi-static motion generation to step over obstacles. Their method relies on a 2D approximation of the robot in which obstacle representation is limited to a rectangle approximation. By taking the zero moment point (ZMP) into account, Stasse et al (2009) were able to extend the method of Yisheng Guan et al (2005); Guan et al (2006) to dynamic movements. 2D line elements are still used for robot and obstacle representation. Arbulu et al (2010) present a method that generates collision-free swing foot trajectories for stepping over quadratic-shaped obstacles. Collision-checking is done at four discrete configurations. Humanoid robots, like our robot LOLA, are able to execute complex stepping motions due to the kinematically redundant structure of their legs. The published methods mentioned above would therefore artificially limit its capacities in complex environments, as complex 3D geometries of the robot's parts and the environment are approximated by line segments or simple boxes. Nor are potential self-collisions taken into account. Horizontal swing foot movements are hence largely restricted and the robot's DoF are therefore not fully exploited. Nonetheless, the presented methods produce remarkable results for the stepping-over motion for one obstacle. However, the viability of the methods has yet to be proven by integrating them into frameworks involving perception and navigation and by validating them in more general environments.

2.2 Robust Walking

There are many different methods to stabilize bipedal walking. We first review two control frameworks that include local modifications as well as an adaptation of future motion. A feedback control framework for the HRP2 biped is presented in Nishiwaki and Kagami (2009b,a). It is based on continuous, moderate frequency recalculation of the walking pattern and local, high frequency adaption of the trajectories. The current state is used as the initial value for solving the trajectory planning problem, which is formulated as a preview control of the linear inverted pendulum model (LIPM). The second stabilization framework is presented in Hirai et al (1998); Takenaka et al (2009), which is applied for the humanoid robot ASIMO. The main feedback variable is the absolute inclination of the upper body, which is treated as horizontal displacement error of the CoM and is used to calculate a reaction moment. This reaction moment aims at restoring an upright posture. Moment regulation is then distributed across three control strategies: *ground reaction force control*, *model ZMP control* and *foot landing position control*. To perform local modifications, Fujimoto et al (1998) introduce a tracking control in task space for the biped's overall dynamics that is based on hybrid position/force control. A walking controller that is based on the full robot model was developed in Loffler et al (2002) for the bipedal robot JOHNNIE. The authors used a feedback-linearization technique to impose linear behavior for the tracking errors. Buschmann (2010) presents a stabilizer that first modifies desired contact forces and torques and then applies hybrid position/force control which generates local task space modifications. Two similar approaches for a whole-body motion controller which also considers long-term stability are presented in Kuindersma et al (2014); Sherikov et al (2014). The stabilizer solves a quadratic programming problem for the overall multibody dynamics at each control step taking, for example, the dynamic constraints of the contact forces and joint torques into account. For long-term stability, the LIPM is used.

Another strategy to stabilize the robot consists in adapting the next footsteps solely based on sensor information in order to adapt its future motion. There are several approaches for the stabilizing step length modifications, such as those that apply heuristics or linear models (for example Pratt et al (2006); Rebula et al (2007); Hodgins and Raibert (1991)). The capture point introduced by Pratt et al (2006) is a method often applied for footstep placement and bipedal walking control (Englsberger and Ott, 2012). Online model predictive control methods to calculate a CoM trajectory and optimize the next footsteps using the LIPM are presented in Urata et al (2011); Tajima et al (2009); Wieber (2006). Urata et al (2011) formulate the optimal control problem for the pendulum differently by choosing the time derivative of the center of pressure (CoP) as input and setting the input's weight in the cost function to zero. This allows an explicit solution to the problem to be determined and more than hundred iterations of the optimization to be computed in each control cycle. To our knowledge, this is the only work that includes an additional step time optimization.

2.3 Versatile and Robust Walking

The authors of Chestnutt and Takaoka (2010) showed experiments with a robot walking on the spot and adjusting footstep locations to reject perturbations while taking obstacles into account. They extended their step planner to compute not only step sequences, but also safe regions around the target footholds. The walking controller uses those regions to calculate permissible

adjustments of the target footholds to reject perturbations in the presence of obstacles. Following this approach, only the step planner takes the environment into account. When the walking controller adjusts the target footholds, it needs to recalculate the robot's trajectories. Collisions can thus only be avoided by using large safety margins, which do not allow considering the possibility for the robot to step close to or over obstacles. Recently, Naveau et al (2017) published a method to extend their walking controller by taking into account convex obstacles as additional constraints. Using the walking controller, the robot *HRP2* can adjust footholds locally to avoid circular obstacles in experiments. They also proved their concept in simulation to reject disturbances. To the best of our knowledge, the combination of perturbations and obstacles has not been shown. Integration into a whole planning framework, including a step planner, is pending. It is a very interesting approach, however, and in contrast to the method presented in this work, the model predictive control with nonlinear constraints is solved at a high frequency. A simplified template model of the robot thus has to be used to meet the real-time requirements. Approximation of the environment and of the robot is kept simple. Complex scenes and self-collisions are therefore difficult to consider in this framework.

The authors of Kuindersma et al (2015) present an optimization based framework that treats all tasks for planning and control in complex scenarios. A step planner calculates valid footholds using mixed integer optimization. It is based on a height map segmented into convex allowable regions represented as polytopes. A human operator has to provide the algorithm seed points to find the polytopes. To include arbitrary environments, they perform the dynamic motion planning with the robot's complete linear and angular momentum equations. This enables multi-contact problems as well as the full kinematics of the robot to be included. For feedback control, a QP is formulated that takes long-term stability into account using the inverted pendulum model. It provides motor commands via additional inverse dynamics for the current time-step. To the authors' knowledge, information about the environment is not used in the stabilization.

The following requirements arise from the shortcomings of the methods described above. The work presented in this paper takes all of them into consideration.

- A complete framework including vision system, step planner and sensor-based trajectory adaptation is presented.
- The framework follows a hierarchical approach where each module takes the results of those preceding it into account.
- Sophisticated models approximate the robot's geometry and dynamics. All computations are performed in real-time.
- The environment can consist of multiple arbitrarily shaped obstacles.
- The method's efficiency will be validated in experiments with complex scenarios, including several simultaneous obstacles and disturbances.

3 System Overview

This section provides a hardware and software overview of the bipedal robot *Lola*.

3.1 Hardware Overview

Our humanoid robot *Lola* has a mass of about 60 kg and is 180 cm tall. It has $n = 24$ position-controlled joints, which are electrically actuated. A detailed view of the kinematic configuration is shown in Fig. 1. Note the kinematically redundant structure of the legs with seven DoF and of the pelvis with two DoF. The robot is equipped with an inertial measurement unit (IMU) in the upper body and 6-axis force-torque sensors (FTS) located in each foot. The IMU consists of three fiber-optic gyroscopes and three MEMS accelerometers. The system includes internal sensor fusion algorithms that provide accurate, drift-free measurements for the absolute orientation and rotation rate. See Lohmeier (2009) for more detailed information. We set up an Asus Xtion PRO LIVE RGB-D camera[1] for environment recognition. It is mounted on a pan/tilt unit on the robot's head. The robot is equipped with two on-board computers: each has an Intel Core i7-4770S@3.1GHz (4x) processor and 8GB RAM. The computer with the vision processing software runs under a Linux OS and the other, on which the walking control executes, runs under a QNX-RTOS. Both computers use TCP to communicate via Ethernet.

3.2 Control Overview

Our control system follows a hierarchical approach that consists of several *modules* (see Fig. 2). Before each walking step of the robot, *ideal motion planning* generates the ideal walking pattern for the next n_{Steps}

[1] ASUS Xtion PRO LIVE, see `http://www.asus.com/Multimedia/Xtion_PRO_LIVE/`

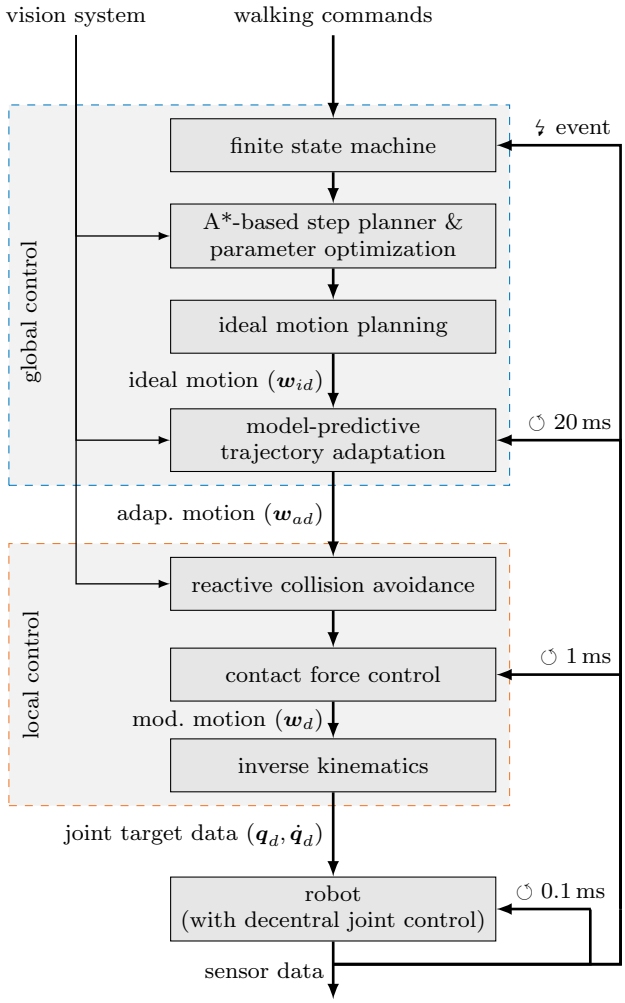


Fig. 2 *Lola*'s real-time walking control system.

steps. The walking pattern consists of the CoP reference trajectories and the ideal task space trajectories $\boldsymbol{w}_{id}(t) \in \mathbb{R}^m$, which are composed of the CoM position, torso rotations, position and orientation of the feet. These trajectories are calculated based on a three-mass model to account for dynamic effects caused by fast leg movements. Buschmann (2010) describes the details. The robot's walking pattern is configured by a parameter set p_{wp}. p_{wp} contains the robot's foothold for each step, parameters describing the robot's movements (e.g. height of center of mass (CoM) or swing foot height), and the step time T_{Step}. p_{wp} is determined by an *A*-based step planner & parameter optimization* (see Section 4) to allow for autonomous navigation and collision-free motion generation in cluttered environments. The key contribution is its real-time capacity. At each walking step of the robot, a new set of p_{wp} is calculated. This allows reactions to user input or dynamically changing environments. Details are explained in Section 4. During step execution, *model-predictive trajectory adaptation* takes external perturbations into account by modifying the desired foot trajectories. The modification depends on the robot's current state provided mainly by the IMU. The output is an adapted motion $\boldsymbol{w}_{ad}(t)$. It is in turn the input to the *local control*, running with a cycle time of $\Delta t = 1\,\text{ms}$. The desired motion at time t_k, $\boldsymbol{w}_{ad,k} = \boldsymbol{w}_{ad}(t_k)$ is modified locally to take sensor input into account. *Reactive collision avoidance* optimizes $\boldsymbol{w}_{ad,k}$ to prevent collisions with obstacles and self-collisions, whereas *contact force control* modifies $\boldsymbol{w}_{ad,k}$ to stabilize the robot. The integration of local collision avoidance is detailed in Section 4, whereas Buschmann et al (2011) presents more details about the stabilization. The modified task space trajectories $\boldsymbol{w}_{mod,k}$ resp. $\dot{\boldsymbol{w}}_{mod,k}$ are input to the *inverse kinematics* at the velocity level (Whitney, 1969; A. Liegeois, 1977), to solve for the joint space velocities $\dot{\boldsymbol{q}}_{d,k} \in \mathbb{R}^n$ from $\dot{\boldsymbol{w}}_{mod,k}$. Since the dimension of the workspace is much smaller than the number of degrees of freedom of the robot, the redundancies can be exploited to minimize a cost function H_y. Thereby, the motion in the nullspace of the robot is determined taking into account self-collision avoidance, constraints related to limits of the joints, and minimization of angular momentum (Schwienbacher et al, 2011; Schwienbacher, 2012). The calculated $\boldsymbol{q}_{d,k}, \dot{\boldsymbol{q}}_{d,k}$ are then passed to the distributed joint controllers. By following this decentralized concept, high sampling rates (50 µs current, 100 µs velocity and position) are reached for the cascaded feedback loops. This way tracking errors of less than 2 mrad can be achieved on all joints. Our methods for *versatile* and for *robust* walking are presented in more detail in the sections below.

4 Methods for Versatile Walking

In this section, we present the strategy developed in the course of our research, which aims at integrating obstacle recognition, collision-free walking in 2D, and whole-body 3D collision avoidance. As will be explained next, different levels of detail in perception as well as in motion generation make real-time navigation possible while exploiting the robots physical capabilities. An exemplary situation is depicted in Fig. 3.

4.1 World Representation

Our approach is based on the representation of both the robot's geometry and the environment via swept sphere volumes (SSVs). This representation allows efficient, accurate geometry approximations and fast distance calculations (Schwienbacher et al, 2011). The robot model,

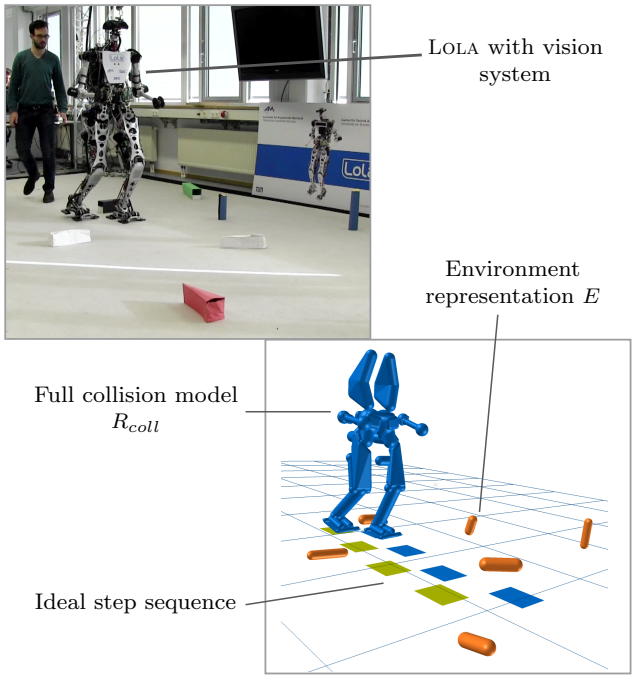


Fig. 3 Left picture: experimental setup with vision system and human disturbing *Lola*; right picture: Lola's collision model, approximated obstacles relevant at current time and calculated step sequence.

R, and the environment model, E, form the approximation of the world $W = \{E, R\}$ used. R and E consist of n_S robot segments and n_O obstacles respectively. Depending on the desired level of detail, each segment or obstacle i is approximated with n_{SSV_i} SSV objects. That way, also non-convex, complex shaped segments can be modeled using multiple SSV objects. Compare, for example, the approximation of the robot's legs (see Fig. 3). In contrast to R, obstacles in E are dynamically added, removed, or modified during run-time.

In order to allow for stepping up on and down from platforms, we added steppable surfaces to our world representation. Surfaces are modeled as convex hulls described via corner points p_c and the normal of the surface. Consistently, the edges of the surfaces are modeled as SSV objects and included in E, since the robot may collide with them. Fig. 4 depicts a surface model.

4.2 Perception

The *vision system* approximates the environment as 3D SSV objects or steppable surfaces. It uses only an onboard RGB-D sensor and updates the objects' positions constantly during walking due to the fast cycle time (approx. 30 ms). This allows the system to track moving obstacles or changes in the perceived environment and

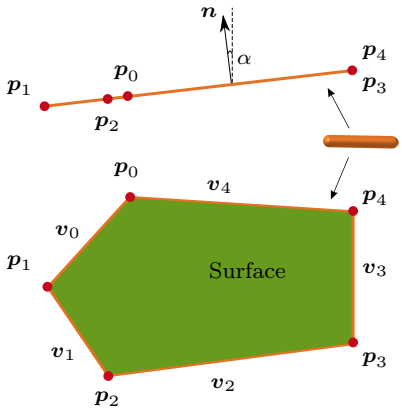


Fig. 4 Surface representation: corner points with edges. Edges modeled as SSV Objects for collision avoidance.

makes it robust against errors such as the drifting of the robot's odometry. Due to real-time requirements, the level of obstacle approximation detail is adapted as a function of the distance to the robot. More details are presented in (Wahrmann et al, 2016).

4.3 *A*-based Step Planner*

At every new physical step taken by the robot, an *A*-based step planner* calculates a sequence of future steps. That way, the robot is able to react to high-level user input or changes in the perceived environment in real-time. The *A*-based step planner* uses a simplified robot model R_{A*} for collision checking with E. It takes into account the lower leg and the foot. The lower leg approximation rotates relative to the foot approximation depending on the analyzed step suggestion. This helps to accurately approximate the robot's motion. It is especially important when taking large steps, both sideways and forward, without the need for large safety margins.

In addition to collision checking, *A*-based step planner* evaluates the pose of the foothold positions with respect to the surfaces. This is efficiently implemented, since it only requires a check to determine whether a foothold, represented by a point, lies in an area described via corner points. The pose of the foothold can be determined by using the normal of the surface.

Furthermore, the *A*-based step planner* determines initial swing foot trajectories for stepping over or swinging past obstacles by changing the corresponding parameter in p_{wp}. It adapts the step time, T_{Step}, based on the desired user input, the robot's stepping movement, and the environment. Further details are published in Hildebrandt et al (2015) and in Wahrmann et al (2017).

4.4 Parameter Optimization

Based on p_{wp}, a three-mass model with simplified kinematics is used to plan the walking pattern. Therefore, the validity of the kinematic movement and possible collisions need to be checked in addition. The *parameter optimization* (see Fig. 2) integrates the movement of the robot's next step using the whole kinematic model. It takes into account the local methods for collision avoidance and, consequently, the approximation of the whole robot's structure for collision checking (R_{coll}) with E. Based on the results of the time integration, p_{wp} is analyzed and optimized. Since the future kinematic movement is analyzed, complex motion generation and advanced error handling for complex scenarios is possible. More details on the specific parameter optimization are published in Hildebrandt et al (2016).

4.5 Reactive Collision Avoidance

In *local control*, the *reactive collision avoidance* locally optimizes the initial solution calculated by the *global control* for avoiding self-collisions or collisions with obstacles. That way, modifications of the robot's movements or the perceived environment due to sensor input can be taken into account. It uses the full robot model, R_{coll}, and modifies all six of the swing foot's DoF as part of $\boldsymbol{w}_{ad}$. The influence of the modifications on the swing foot's trajectories on the robot's stability can be compensated, since the robot has a redundant kinematic configuration and the real CoM is tracked – not the torso position. More details are presented in Hildebrandt et al (2014).

5 Methods for Robust Walking

To enable the biped to react to unknown disturbances, we introduced a *model-predictive trajectory adaptation*. The ideal planned motion $\boldsymbol{w}_{id}(t)$ is adapted in the *global control* module based on current sensor data. This is done with a prediction model which allows trajectory modifications to be optimized in real-time for a certain time horizon. This way, the robot's adapted motion takes robot dynamics (in a simplified manner), desired motion, and current state into account. We choose to modify the leg trajectories by changing final values for the x–,y– and z–position as well as the final horizontal orientation (Wittmann et al, 2015b) for the next step. Stabilizing motions for more than one step could be included, but will increase computational time drastically. The horizontal CoM trajectories can furthermore be adapted (Wittmann et al, 2016). The main goal of the presented method is to stabilize the robot's absolute inclinations with regard to the ground. For stiff position-controlled robots, they are considered to be the main DoFs that deviate from the ideal values during disturbances. To estimate these quantities, the IMU's measurement data is filtered in a state observer (Wittmann et al, 2015a).

5.1 Dynamic Prediction Model

We choose to use a planar dynamic model (Fig. 5) of the biped that directly includes the inclinations mentioned (Wittmann et al, 2014). The full multi-body model of a bipedal robot is simplified as follows: the planned CoM and foot trajectories are assumed to be perfectly tracked in the robot's planning frame of reference (FoR) which rotates with the robot (T-system). Analogous to the model used in *ideal motion planning*, inertia effects are approximately included via three point masses. We use the same planar model for the x- and y-directions (the motion in the sagittal and frontal plane, respectively) with different geometry and trajectories. The following description is related to the x-direction. The unactuated DoFs are the inclination φ_x in the x-direction and the vertical translation z (translation in x-direction is neglected). They describe the transformation from an inertial (index I, x_I, z_I) to a FoR rotating with the robot's upper body (index T, x_T, z_T). Furthermore, the ideal planned trajectories determined by the walking pattern generation in the T-system correspond to the trajectories for the robot's upper body ${}_T\mathbf{r}_b(t)$ and the feet ${}_T\mathbf{r}_{f1}(t)$, ${}_T\mathbf{r}_{f2}(t)$. The upper body and the feet are approximated via point masses and the contact between foot and ground is approximated as one point contact located at the middle of the foot. The relative position of the three masses is constrained to follow a known trajectory determined by the planner. The contacts are linear spring-damper pairs (stiffness k_c, damping d_c) with the values identified from the real robot's rubber sole. They act only in z_I-direction and are considered to be unilateral. The overall equations of motion (EoM) for the model with $\boldsymbol{q} = [z,\, \varphi]^T$ can be stated in the form

$$\boldsymbol{M}_p(\boldsymbol{q}, t)\ddot{\boldsymbol{q}} + \boldsymbol{h}_p(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = \boldsymbol{\lambda}_p(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) + \boldsymbol{T}_s \tag{1}$$

where $\boldsymbol{M}_p$ is the mass matrix of the prediction model, $\boldsymbol{h}_p$ contains the nonlinear terms, and $\boldsymbol{\lambda}_p$ represents the resultant force and moment of all active contacts projected on the model's DoFs. $\boldsymbol{T}_s$ describes an additional stabilization torque which is calculated with a saturated PD controller. It adds the robot's contact force control to the model. Due to the rotation with φ, the EoM are nonlinear but can be stated analytically.

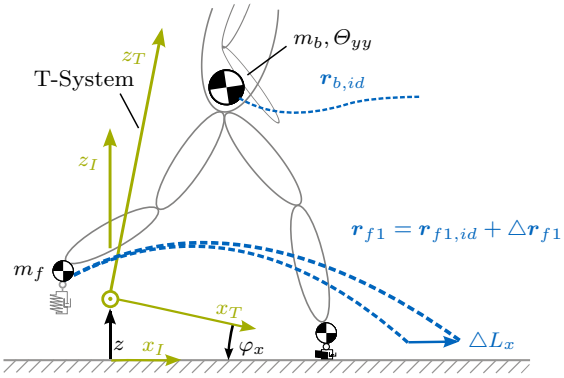


Fig. 5 Prediction model with three point masses and swing foot modification Δr_{f1}.

5.2 State Estimation

The algorithm requires an initial value $\boldsymbol{z}_0 = [\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0]^T$ of the prediction model for the robot's current state. It is obtained by feeding the IMU measurements, namely the absolute inclination $\boldsymbol{\varphi}_m$ and inclination rate $\dot{\boldsymbol{\varphi}}_m$, into a state observer. The observer is similar to the one presented in Wittmann et al (2015a), except that it uses the nonlinear model (1). It is based on an extended Kalman filter with compensation for model errors and external disturbances. The observer is implemented for both directions and provides an initial state for the model in the x- and y-directions $\boldsymbol{z}_0$.

5.3 Model-Predictive Trajectory Optimization

The following description only treats footstep modifications, but the method can be extended to CoM modifications as shown in Wittmann et al (2016). A parameter ΔL_x which describes a horizontal displacement of the final swing foot position is introduced. It is used to calculate a quintic polynomial that begins at the current position, velocity and acceleration and ends at ΔL_x with zero velocity and acceleration. This is shown in Fig. 5, where the polynomial $\Delta_T \boldsymbol{r}_{f1}(\Delta L_x)$ is added to the ideal trajectory ${}_T\boldsymbol{r}_{f1,id}$ via

$$ {}_T\boldsymbol{r}_{f1} = {}_T\boldsymbol{r}_{f1,id} + \Delta_T \boldsymbol{r}_{f1}(\Delta L_x). \tag{2} $$

This consequently changes the contact force's lever arm, thereby influencing the state of the model (1). The overall first order differential equation with the additional parameter yields

$$ \begin{aligned} \dot{\boldsymbol{z}} &= \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}_p^{-1}(\boldsymbol{q},t)[\boldsymbol{\lambda}_p(\boldsymbol{q},\dot{\boldsymbol{q}},\Delta L_x,t) + \boldsymbol{T}_s - \boldsymbol{h}_p(\boldsymbol{q},\dot{\boldsymbol{q}},t)] \end{bmatrix} \\ &= \boldsymbol{f}(\boldsymbol{z},t,\Delta L_x). \end{aligned} \tag{3} $$

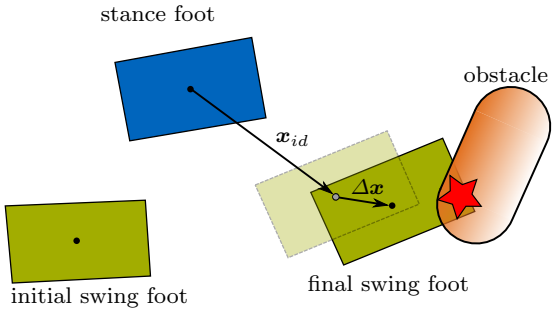


Fig. 6 Example of problem description: ideal collision-free final swing foot position and the modified invalid position.

The presented stabilizer's goal can be formulated mathematically as a minimization of a quadratic cost function

$$ J = \Delta \boldsymbol{z}^T(t_f)\boldsymbol{S}_z\Delta \boldsymbol{z}(t_f) + s_p\Delta L_x^2 + \int_{t_0}^{t_f} \Delta \boldsymbol{z}^T \boldsymbol{Q} \Delta \boldsymbol{z}\ \mathrm{d}t \tag{4} $$

for a certain time horizon $t \in [t_0, t_f]$. Note that the cost function weights, $\boldsymbol{S}_z$ and $\boldsymbol{Q}$, are mainly parameterized to minimize the absolute inclination error φ_x over a certain time horizon, which is included in the state error $\Delta \boldsymbol{z} = \boldsymbol{z} - \boldsymbol{z}_{ref}$. $\boldsymbol{z}_{ref}$ is the reference state in a perfect upright posture. The optimization problem is solved using a direct shooting method. The model (3) can be solved numerically for a given initial value $\boldsymbol{z}(t_0) = \boldsymbol{z}_0$ and ΔL_x. The resulting trajectory $\boldsymbol{z}(t, \Delta L_x)$ can then be used to compute the cost function $J(\boldsymbol{z}(t, \Delta L_x), \Delta L_x)$. This way, the problem is converted into the static optimization problem

$$ \min_{\Delta L_x \in \mathcal{A}_s} J(\Delta L_x) \tag{5} $$

which is unconstrained as long as the variable ΔL_x remains inside the allowable set $\mathcal{A}_s$. It can be solved with nonlinear programming methods such as Newton's method (Nocedal and Wright, 2004). The set $\mathcal{A}_s$ can be interpreted geometrically as a *valid area*. It takes into consideration constraints due to the robot's kinematics as well as restrictions resulting from obstacles located near the robot. The determination of $\mathcal{A}_s$ and its integration in the optimization is the subject of the next section.

6 Versatile and Robust Walking

The methods presented are now combined to obtain *versatile and robust* walking simultaneously. In keeping with the hierarchical control system architecture, we decide to first determine a collision-free motion that is then adapted for disturbance rejection based on sensor

data. Furthermore, we define that avoiding a collision has a higher priority than rejecting a disturbance. The decision is based on the assumption that disturbance rejection can take place during more than one step, whereas a collision will lead to a system failure. This means that the possible solutions of (5) are restricted to reachable and obstacle-free regions. Fig. 6 shows an exemplary situation which is addressed below.

We start from a valid final swing foot location $\boldsymbol{x}_{id}$, determined by the *A*-based step planner* taking into account E. Assuming an unknown disturbance, *trajectory adaptation* then modifies this position by $\Delta\boldsymbol{x} = [\Delta L_x, \Delta L_y]^T$ to stabilize the robot. The resulting final foothold position, however, would cause a collision. The main question now is how such constraints can be described and accounted for in a real-time optimization procedure. Short computation time is crucial, since the method is used with sensor feedback and has to react instantaneously to unknown disturbances. Longer calculation times would incur in higher latencies and, therefore, would degrade the performance of the feedback control. The method's desired output is a modified yet collision-free foot position

$$\boldsymbol{x}_m^* = \boldsymbol{x}_{id} + \Delta\boldsymbol{x}^*. \tag{6}$$

with feasible modification $\Delta\boldsymbol{x}^*$ instead of the modification $\Delta\boldsymbol{x}$ which could lead to collisions.

There are basically two approaches to handle such situations: The first is to determine an optimal step length modification without constraints and project the final solution onto the cone of $\mathcal{A}_s$. The second accounts for the constraints during optimization, however, this requires an optimization for the step length modifications in both directions since the feasible set is at least two-dimensional and the boundaries for ΔL_x and ΔL_y are coupled. For this reason, we extend the decoupled planar models presented in Subsection 5.1 to a single 3D model. The problem description is summarized as follows:

- Short calculation time ($\ll 1\,\mathrm{ms}$)
- Several arbitrarily shaped obstacles
- Stepping over of obstacles should be possible
- Kinematic limits must be checked
- A collision-free position must be found reliably

The constraints for such situations are described below. Projection of an invalid point onto a feasible region is discussed in the subsequent part. This projection method is then applied to the stated problem and two solution strategies are discussed.

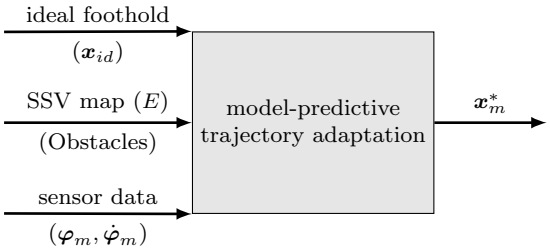


Fig. 7 Overview of the main input and output data for the sensor-based trajectory adaptation with additional obstacle avoidance.

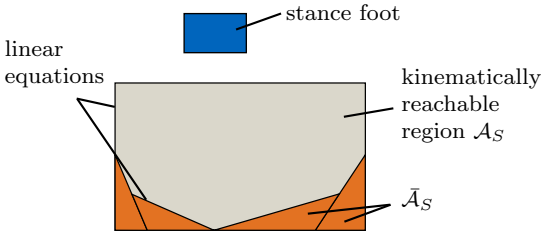


Fig. 8 Kinematically reachable region (gray) and current stance foot (blue). Kinematically reachable region bounded by seven linear equations.

6.1 Geometrical Constraints

The key point when taking into account the geometrical constraints imposed on robust walking, namely kinematic constraints and obstacles, is their consistent and dense representation. To allow fast calculations, we chose convex polytopes to represent both kinematic constraints and obstacles. This results in n_{eq} linear inequalities,

$$c_{eq,j} := \boldsymbol{a}_j^T \boldsymbol{x} > b_j. \tag{7}$$

A set of n_{C_I} linear inequalities, restricted by the corresponding corner points, describes one convex polytope, $\mathcal{C}_I$, which is an *invalid region*. In total, the *invalid area* $\bar{\mathcal{A}}_S$ consists of n_p polytopes and is defined as follows:

$$\bar{\mathcal{A}}_S := \bigcup_{i=1}^{n_p} \mathcal{C}_{I_i}. \tag{8}$$

The corresponding *valid area*, $\mathcal{A}_S$, is formally defined as

$$\mathcal{A}_S := \mathcal{A} - \bar{\mathcal{A}}_S, \tag{9}$$

based on the total search area $\mathcal{A}$. The determination of $\bar{\mathcal{A}}_S$ is described below.

6.1.1 Kinematic Limits

Starting with the current stance foot, the kinematically reachable area is approximated by the convex polytope

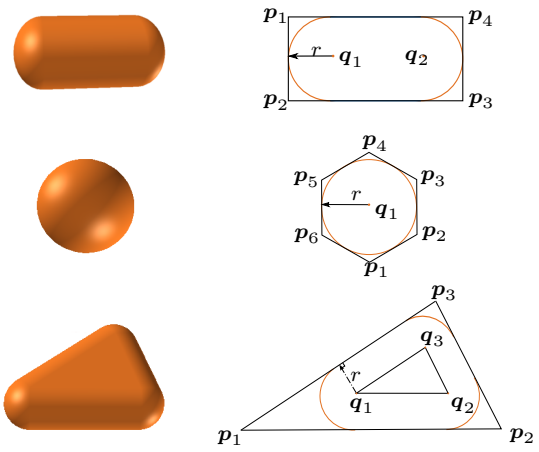


Fig. 9 Approximation of SSV Objects as polytopes.

as depicted in Fig. 8. The unreachable area results in n_k convex polytopes (see Fig. 8). The system of inequalities for one convex polytope results in

$$\mathcal{C}_{I,k,j} = \{\boldsymbol{x} | \forall i \in \{1, ..., n_{k,j}\} : \boldsymbol{a}_{k,j,i}^T \boldsymbol{x} > b_{k,j,i}\}. \quad (10)$$

6.1.2 Obstacles

As described in Section 4, 3D segments approximate obstacles and areas of the environment the robot cannot step onto. The former consist of n_{SSV} convex SSV objects to allow for a detailed approximation of the real objects. Like the representation of the kinematic limits, each of the $n_{SSV,i}$ convex SSV objects are reduced to 2D polytopes to comply with the hard timing constraints.[2] The SSV objects are first projected onto the ground. Then the three types of SSV objects – sphere, line and triangle – are represented as polytopes as shown in Fig. 9. For each object j, we get a convex hull, which is described by

$$\mathcal{C}_{I,SSV,j} = \{\boldsymbol{x} | \forall i \in \{1, ..., n_{eq,SSV,j}\} : \boldsymbol{a}_{SSV,j,i}^T \boldsymbol{x} > b_{SSV,j,i}\}. \quad (11)$$

Kinematic limits already restrict the valid area for a step. Therefore, only obstacles within this kinematically reachable area are considered. This approach greatly reduces the computational costs for the inequality constraints.

[2] Note, one arbitrary shaped obstacle may be approximated by $n_{SSV,i}$ convex SSV objects. Here, we reduce all SSV objects of all n_O obstacles to 2D polytopes without taking into account their belonging to same obstacles. That way, we get only 2D polytopes representing the total of E.

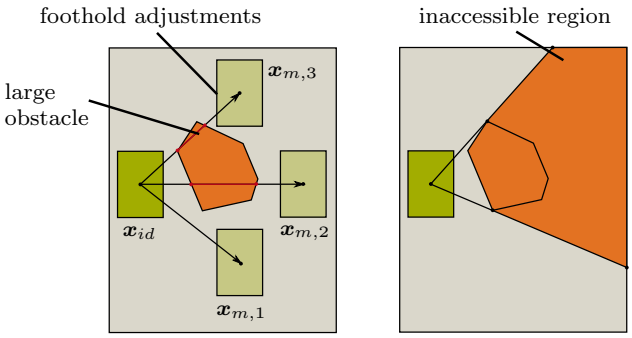


Fig. 10 Large obstacle - Representation of inaccessible regions.

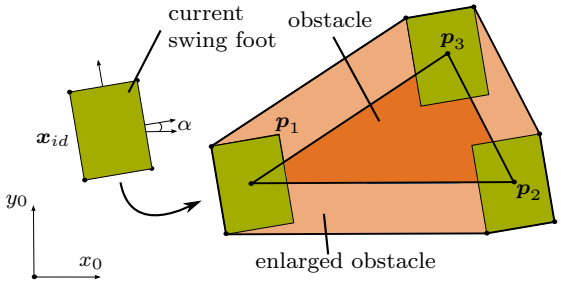


Fig. 11 Obstacle enlarged by foot geometry.

6.1.3 Large Obstacles

Considering the robot's whole kinematic movement, obstacle-free regions are not necessarily steppable or kinematically reachable. Large obstacles, which the robot cannot step over because of their height, make adjacent obstacle-free regions inaccessible due to kinematic constraints. We already consider these large obstacles in the environment representation to avoid repeated checks for obstacle-free regions and inaccessible regions. Instead of introducing a polytope for the inaccessible region, the large obstacle's representation is enlarged (refer to Fig. 10). That way, the obstacle approximation as well as the representation of the inaccessible region stay convex.

6.1.4 Foot Geometry

The sophisticated 3D representation of the foot and lower leg geometry used in the *A*-based step planner* helps to give a better approximation of the robot's whole kinematic movement. We nevertheless enlarge the obstacles by the foot geometry as shown in Fig. 11. Since the foot orientation does not have a large influence on perturbation rejection, the current robot's foot orientation α, calculated by the *A*-based step planner*, is thereby taken into account. Thus the geometric constraints can be analyzed using only one point, which describes the foothold position.

6.2 Finding Safe Footholds

Below, we try to answer the question initially asked: Given a modified but invalid foothold position, how can the closest valid one be determined (see Fig. 6)? Three methods are described and compared in the following.

6.2.1 Sampling

One answer is to discretize the area around the modified foothold position. The discrete positions are checked to determine whether they are valid or not. From among all the valid positions, the one closest to the modified position is chosen. The advantage of this procedure is that calculation time only depends on the number of equations and on the discretization level. The calculation time is therefore deterministic, making the method suitable for real-time application. Furthermore, all discretized points can be checked in parallel on multiprocessor platforms. The disadvantage is the strong dependence between the solution's quality and the discretization grid. Fine discretization is computationally expensive, but is necessary to find a solution in complex scenarios.

6.2.2 Geometric Testing

Another solution is to search for the closest valid point on the boundaries, which are described by linear equations. This reduces the problem to a distance calculation between this valid point in $\mathcal{C}_{I_i}$ and line segments describing the boundaries of $\mathcal{C}_{I_i}$. The closest point on the line segment can be calculated based on the smallest distance between point and line segment. To accelerate the calculations, the residuum of the point and the inequalities is used: A small residuum indicates a small distance between the point and the boundaries and is used to reduce the number of distance calculations. However, this procedure is not possible for multiple invalid regions that intersect each other. The closest point on one of the boundaries is not necessarily valid since it may lie in another invalid region. This extended problem is solved as follows: (1) The closest point on all boundaries has to be found. (2) Each of these points has to be checked for validity. If one is invalid, the next nearest valid point on the same boundary is chosen. (3) Finally, all resulting points have to be compared to find the closest point. In contrast to the *sampling*-based approach, the quality of the solution does not depend on the environment or prior assumptions (e.g., a discretization grid). However, the closest point is searched for on each boundary and thus the computational costs increase significantly with the number of invalid regions.

6.2.3 Safe Regions

The previously presented solutions all begin with an invalid starting point, $\boldsymbol{x}_m$, and try to find a valid point, $\boldsymbol{x}_m^*$. Another possibility is to invert the problem. This idea comes from computer graphics and uses a decomposition of the valid region (Greene, 1983): instead of finding the closest valid point, $\mathcal{A}_S$ is divided into a set of convex valid regions C_V. In each of these valid and convex regions, the closest point to the invalid starting point can be determined separately and, because of the regions' convexity very efficiently. In contrast to the *Geometric Testing* method, intersections of the valid convex regions do not pose a problem, because the search is applied on a set of valid regions. The solutions are consequently independent of each other. The best one is chosen based on the calculated set of closest points. This procedure of searching valid convex regions instead of only considering the invalid regions is largely inspired by Chestnutt and Takaoka (2010), who presented a method to calculate a valid convex area around a valid starting point. The algorithm starts with a convex area. It iterates around the starting point and it removes invalid parts of the initial convex area. A drawback of the implementation is that only one convex area around the starting point is found. Deits and Tedrake (2015) presented a powerful open source tool, called *IRIS*, which has already been applied to step planning (Deits and Tedrake, 2014). *IRIS* uses the corner points of invalid convex regions as input and calculates the corresponding inequalities. It also determines the largest valid convex region which is closest to a starting point. Although it seems to be well suited for the present problem, it exhibits some shortcomings with respect to our application[3]:

- *IRIS* does not guarantee that the starting point lies in the convex region.
- *IRIS* needs a predetermined search area. The algorithms seem to be very sensitive to starting points close to the borders.
- *IRIS* only determines one convex area. When it is applied iteratively on the same search area by respectively removing the determined regions, it often fails to find additional valid regions.

Sarmiento et al (2005) and Liu et al (2010) present methods to divide arbitrary areas in convex regions. On the one hand, neither method is restricted to convex invalid regions; however, neither benefits from reduced computational costs for convex problems. In our application, only convex invalid regions are used, to obtain

[3] The following statements are based on the authors' experience with the *IRIS* tool, which is available as open source.

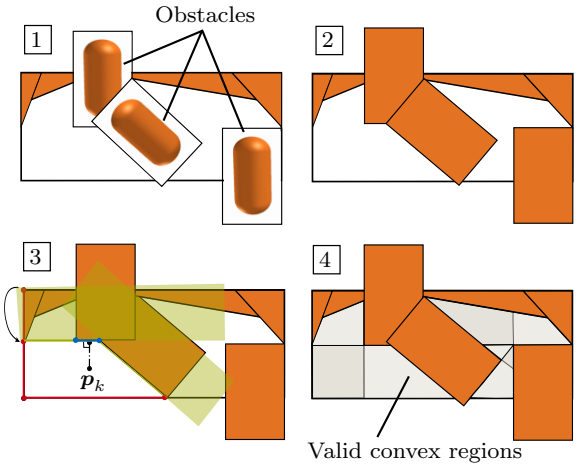


Fig. 12 Finding convex valid regions. (1) and (2) obstacles and kinematic constraints. (3) Seed point p_k. Already found inequalities in green. Current closest line in blue. Remaining active boundaries in red. (4) Output of Algorithm; Multiple convex valid regions (ivory).

the benefit in terms of calculation times. Like Sarmiento et al (2005) and Liu et al (2010), we want to completely cover the valid area with multiple convex regions that might intersect. We exploit the following characteristics of this mathematical problem to meet our timing limitation:

- $\mathcal{C}_{I,i}$ are all convex.
- Our algorithm's starting point, which is the ideal step location, $\boldsymbol{x}_{id}$, calculated by A*-based step planner, always lies in $\mathcal{A}_S$.
- The kinematic constraints limit $\mathcal{A}$.

Our algorithm takes an iterative approach. We set up a grid of seed points over $\mathcal{A}$. The seed points, which lie in $\mathcal{A}_S$, successively become the starting points for searching a convex region surrounding each. Once the convex region around a seed point is determined, it is removed from the remaining valid area to avoid repetitive searches. Since it is valid per definition, we use the ideal footstep location as a first seed point. This helps to reduce the computational costs. The number of seed points is set to a value which ensures that no valid regions are missed in the experiments. For one seed point, the search for a valid convex region can be summarized as follows: The closest boundaries to the seed point are determined iteratively. The boundaries are therefore considered to be line segments with start and end points. Once the closest boundary, l_j, is found, it is added as a linear inequality to the valid region $\mathcal{C}_{V_k}$. Here, the convexity of the invalid region is used to efficiently find the closest boundaries. All boundaries outside $\mathcal{C}_{V_k}$ are skipped and considered inactive for the following steps (see Fig. 12). The search stops when there are no active boundaries left. The algorithm is summarized in Algorithm 1. The calculation of $\mathcal{C}_V$ has to be done only once before each of the robot's physical steps. The kinematically reachable area lies outside the camera's field of view. Therefore, the representation of the environment does not change during the execution of the step.

Algorithm 1 Dividing A_S in convex regions

function FIND-CONVEX-REGIONS($\mathcal{A}_S$, $\bar{\mathcal{A}}_S$)
  initialize set of seed points $\mathcal{P}_S$
  $\mathcal{C}_V = \{\}$
  for all $p_k \in \mathcal{P}_S$ **do**
    verify p_k valid?
    j $\leftarrow 0$
    repeat
      Calculate closest boundary line l_j of $\bar{\mathcal{A}}_S$
      Add l_j as inequality to boundaries of $\mathcal{C}_{V_k}$
      Remove all inactive boundaries of $\bar{\mathcal{A}}_S$
      $j \leftarrow j + 1$
    until no more active boundaries
    remove $\mathcal{C}_{V_k}$ from $\mathcal{A}_S$
    update $\mathcal{C}_V = \mathcal{C}_V \cup \mathcal{C}_{V_k}$
  end for
  Output: $\mathcal{C}_V$
end function

6.2.4 Discussion

In summary, all of the methods presented can be used to find a valid point which is close to the desired invalid point. They have been evaluated in simulation. In our implementation, the *sampling*-based method showed strong dependence on the grid size. The *Geometric Testing* method and the *Safe regions* method showed similar results in terms of computational costs and quality of the solution. However, the latter has the advantage that it calculates not only the closest point, but also $\mathcal{C}_V$. These inequalities are suitable for optimization algorithms as shown below. We therefore chose the *Safe Regions* method for the final implementation on the robot.

6.3 Footstep Modification with Geometrical Constraints

Finally this section describes how the algorithm for finding a point in the safe regions can be combined with the optimization of the next foothold positions (see Subsection 5.3) for stabilizing the robot.

6.3.1 Projected Optimization Result

One straightforward solution for considering safe regions is to project the optimized quantities $\Delta\boldsymbol{x} =$

$[\Delta L_x, \Delta L_y]^T$ onto the safe regions. This way, the step length modifications can be optimized separately in the x- and y-directions. The optimization results are projected onto the cone of $\mathcal{A}_S$. We tried two different criteria to find the point that is closest to the optimal solution: the geometric distance between $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{x}^*$ and the point with the best (lowest) costs determined using (4). For the second criteria several candidate points are generated, all lying on the cone of $\mathcal{A}_S$. Nevertheless, this requires additional time-consuming evaluations of (3) and (4). We tried both methods with the result that they perform similarly and consequently chose the closest distance criterion.

6.3.2 Optimization with Inequality Constraints

A mathematically exact way is to include the inequality constraints directly in the optimization. This can be realized with hard constraints or using a penalty function. The latter has the advantage that the problem is again unrestricted and exhibits better computational time. We extend the prediction model (3) by an additional passive DoF, φ_y, which is an inclination in y-direction. The foot trajectory

$$ {}_T\boldsymbol{r}_{f1} = {}_T\boldsymbol{r}_{f1,id} + \Delta\, {}_T\boldsymbol{r}_{f1}\,(\Delta L_x, \Delta L_y) \tag{12} $$

includes both final foot step modification quantities ΔL_x and ΔL_y. The EoM of the spatial model with $\boldsymbol{q}_s = [z, \varphi_x, \varphi_y]^T$ can be derived and written as first order differential equation

$$ \dot{\boldsymbol{z}}_s = \boldsymbol{f}_s(\boldsymbol{z}_s, t, \Delta L_x, \Delta L_y) \quad , \boldsymbol{z}_s = [\boldsymbol{q}_s, \dot{\boldsymbol{q}}_s]^T. \tag{13} $$

The cost function for the optimization is rewritten for the spatial model (13) as

$$ J_s = \boldsymbol{z}_s^T \boldsymbol{S}_z \boldsymbol{z}_s + \Delta\boldsymbol{x}^T \boldsymbol{S}_p \Delta\boldsymbol{x} + \int_{t_0}^{t_f} \boldsymbol{z}_s^T \boldsymbol{Q} \boldsymbol{z}_s \mathrm{d}t + h(\Delta\boldsymbol{x}) \tag{14} $$

and extended by an additional penalty term

$$ h(\Delta\boldsymbol{x}) = \begin{cases} \beta\,(\boldsymbol{x}_m - \boldsymbol{x}_p)^2 & \boldsymbol{x}_m \in \bar{\mathcal{A}}_S \\ 0 & \boldsymbol{x}_m \in \mathcal{A}_S \end{cases} \tag{15} $$

that includes the distance from $\boldsymbol{x}_m = \boldsymbol{x}_{id} + \Delta\boldsymbol{x}$ to the closest valid point $\boldsymbol{x}_p$ at the cone of $\mathcal{A}_S$. The additional weight β is set to a value higher than all other weighting matrix entries. The optimization result from (13) and (14) is not necessarily valid, since invalid solutions are penalized but not completely avoided. Nonetheless, the solution is at least close to $\mathcal{A}_S$. Consequently, the validity of the optimization result, $\boldsymbol{x}_m$, will be verified and if necessary projected onto $\mathcal{A}_S$ as described in Subsection 6.3.1. In simulation and experiments, this strategy shows the ability to find solutions that require stepping over of obstacles to maintain the robot's balance. This is possible because instead of using just one safe region, a set of safe regions is used. Instead of using always the closest point as the starting point for the optimization, we could use starting points in each of the safe regions as well. That way, we could run multiple optimization processes in parallel, and chose, as the final result, the optimal result out of all optimization processes. So far, this approach has not shown superior results to just using the closest point as the starting point. The same extension is applicable for the *Footstep Modification with Geometrical Constraints* method.

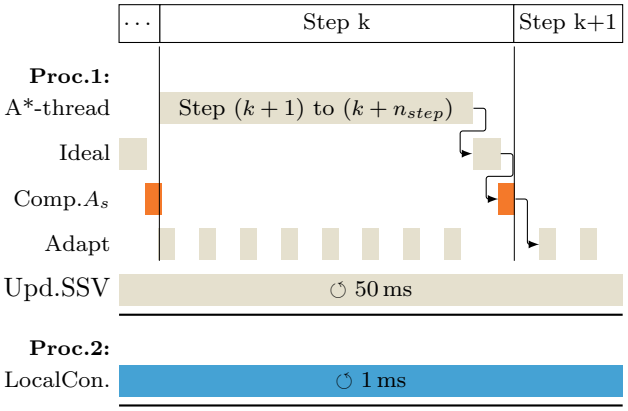


Fig. 13 Multi-process and multi-thread software architecture of LOLA's real-time walking control system.

6.4 Implementation Details

The following paragraph provides more details about the implementation and real-time realization of the presented framework. Since several computationally expensive algorithms have to be solved, the overall control system is split into three main processes. Fig. 13 shows an overview. Process 1 includes global planning and has an additional thread for the *A*-based step planner* and the *parameter optimization* (*A*-thread*) to compute the step sequence for the following $(k+1)$ to $(k+n_{step})$ steps. After the A*-search and before a new step k begins, the ideal motion is planned (*Ideal*) and the set of valid regions $\mathcal{A}_S$ is computed from the SSV map. The map is continuously updated with data from the vision system (*Upd.SSV*). While the robot executes its current step, the model-predictive trajectory adaptation (*Adapt*) uses sensor data to modify the ideal motion according to the strategy detailed in Section 5 and Sec-

tion 6. Local control runs in a second process (LocalCon.), while a third process (not shown) performs low-level communication with the EtherCAT bus as well as sensor and target data exchange. Communication between the different processes is realized via the system's shared memory. Both the local control and EtherCAT bus run with a cycle time of 1 kHz.

$\mathcal{A}_s$ is computed only once before a new physical step. This is sufficient since obstacles that are within the kinematic limit of one step do not lie inside the camera's field of view and and, therefore, are no no longer updated. *Reactive collision avoidance* prevents self-collisions of the robot or collisions of the robot with obstacles that a footstep adaptation might cause due to modified motion kinematics. This may otherwise occur since only the final position is checked and the swing foot height is adapted via a heuristic.

Table 1 Computational time summary for maximal 4 obstacles. Runtimes are obtained from the real-time QNX computer.

method	avg. [µs]	max. [µs]
Comp.$\mathcal{A}_S$	600	2000
Find closest point	4	250
Adapt	1000	2500

7 Results

The videos for the presented results can be found on the project's website as well as on our YouTube channel.[4]

7.1 Further Results

In this work, we presented the overall framework for *versatile and robust walking*. The framework was validated in multiple environments including vision-based stepping up and down platforms (see `https://youtu.be/rKsx8HKvBkg`), moving obstacles (see `https://youtu.be/-VvxzFg9ATU` and `https://youtu.be/6PLN6B4vSHM`), and different perturbations (see `https://youtu.be/46YIWkYWuoY`). Furthermore, it was presented in several public demonstrations (see, for example, `https://youtu.be/g6UACMHgt20`). Due to the limited space, we focused in this article on experiments with explicit disturbances, such as external pushes in the presence of obstacles. For the robot to succeed in these test cases, the methods presented in this paper need to act simultaneously. This makes the test cases on the one hand more complex for the methods and on the other hand more complex for the authors to set up experiments which show the influence of all methods.

7.2 Simulation

Simulation results for the walking control framework described above are presented below. The control system is implemented for LOLA's multi-body simulation. The simulation environment takes unilateral and compliant contacts, motor dynamics, as well as the joint control loops into consideration. The simulation example is a simple synthetic one, since it is used to show the algorithm for footstep adaptation with one additional obstacle. The humanoid is commanded to walk in place while it is subjected to a disturbance force in its walking direction (x-direction). One obstacle is placed close to the robot to limit feasible footstep modifications. Fig. 14 shows the disturbance force and the resulting inclination errors for the constrained optimization result (Subsection 6.3.1). The robot is still able to stabilize itself with the limited foot positions (Fig. 15). A snapshot of each walking step of the 3D collision model and the 2D polytopes is shown in Fig. 16. An explanation of the 2D plot is given in Fig. 17. The same simulation experiment was conducted with the 3D prediction model, including the inequality constraints during optimization (Subsection 6.3.2). Results are not presented separately since they closely resemble those shown above. Due to its reduced computational time, we decided to test the footstep optimization with the restriction of the optimization result on the real robot.

7.3 Experiment

Two experiments will be presented to show the proposed method's effectiveness under real-world conditions. The first experiment is a synthetic case resembling the simulation case, whereas the second exhibits a more general setup. In experiment 1, the obstacle is put right in front of the robot. In contrast to experiment 2, the obstacle is manually approximated without input from the vision system according to the simulation setup (see Fig. 21). We do not use the vision system for two reasons: (1) According to the simulation setup, the obstacle is not in the camera's field of view. (2) This experiment should examine only the procedure for the method to consider obstacles during disturbance

[4] Project's website: `https://www.amm.mw.tum.de/en/research/current-projects/humanoid-robots/walkinguneventerrain/`. Videos presenting the results: `https://youtu.be/RjqAh3Blxng`

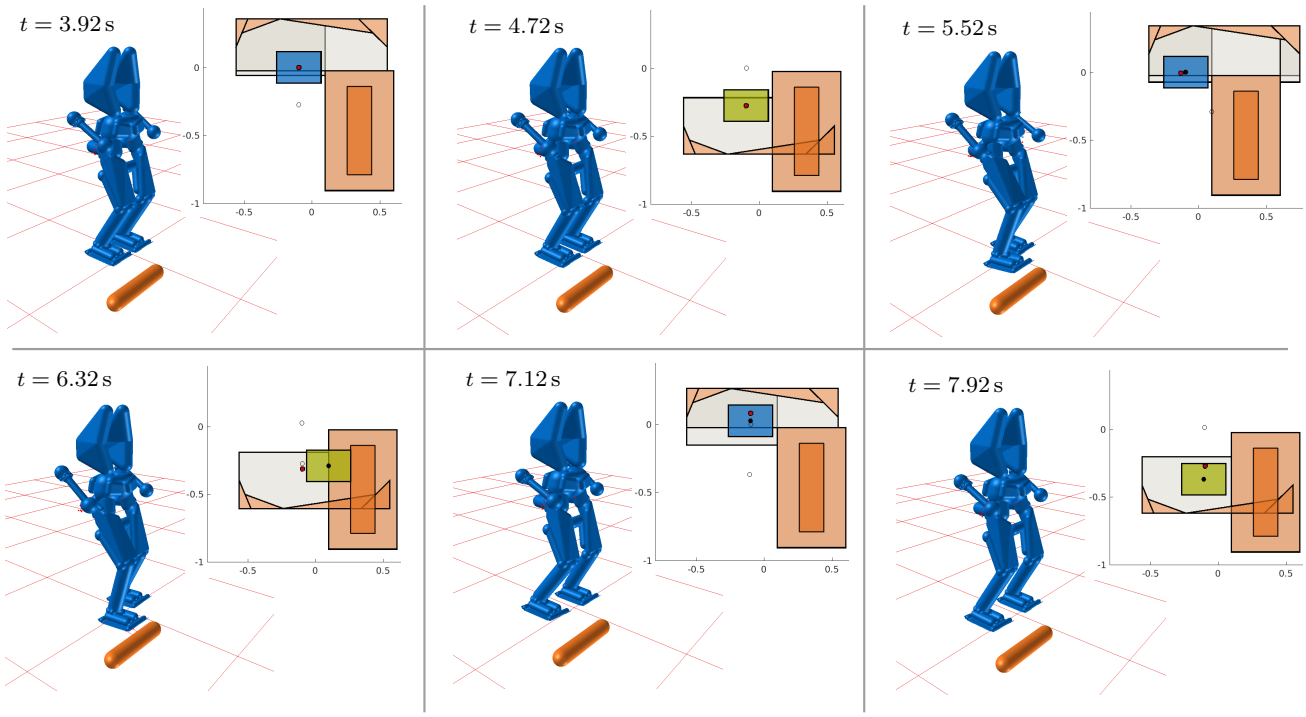


Fig. 16 Synthetic simulation example: collision model and footstep adaptation with one obstacle at different time instants. The robot is stepping in place and a forward disturbance force is given around $t = 5\,\mathrm{s}$. Distance units in m.

rejection, without having the uncertainties of a running vision system. The sequence in Fig. 21 shows the robot at different time instances after it is pushed from behind. The corresponding modification trajectories for the feet and inclination errors of the upper body are depicted in Fig. 18 and Fig. 19. The limitation of the footstep modification, ΔL_x, can be seen in Fig. 20. The obstacle lying in front of it limits the right foot's step (second picture in Fig. 21). In the next step, the robot performs a huge step modification to avoid divergence of the inclination error.

Experiment 2 also includes the vision system. The setup is presented in Fig. 23. The robot is commanded to walk forward with 30 cm step lengths. While walking, the robot's walking control registers the obstacles detected online. The *A*-based step planner & parameter optimization* module calculates in real-time an ideal step sequence and parameter set that ensure collision-free movements. The ideal motion is modified based on the robot's state. The robot is pushed several times during the experiment and recovers from the disturbances. The overall ideal step sequence and modified footholds calculated are shown in Fig. 22.

8 Discussion

8.1 Summary

In this paper, we propose strategies for *versatile* and *robust* walking and present their integration into an overall hierarchical framework. Control of the robot is divided into modules. Each module takes into account and improves the result of the previous one. This way, the whole system becomes robust. The motion planning problem is furthermore divided into smaller parts that can be solved efficiently in real-time. We moreover show how methods for *versatile* and *robust* walking not only can coexist, but can cooperate to achieve *versatile and robust* walking. We develop a technique to efficiently transform our environment model to a representation by inequalities. These inequalities are included as constraints in the optimization of footstep modifications for disturbance rejection. We present and discuss several solutions to solve the resulting constrained optimization problem in real-time. We analyze and validate our approach in simulations and experiments.

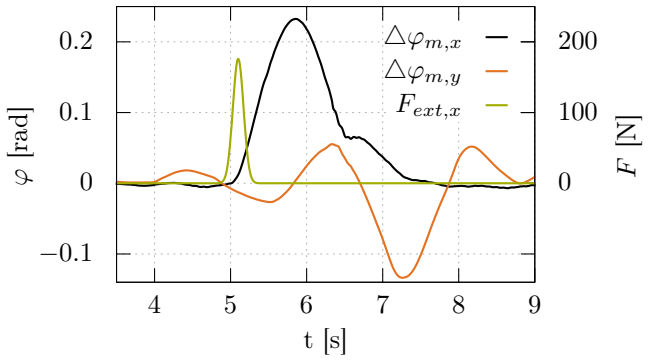


Fig. 14 Synthetic simulation example: disturbance force and resulting inclination errors.

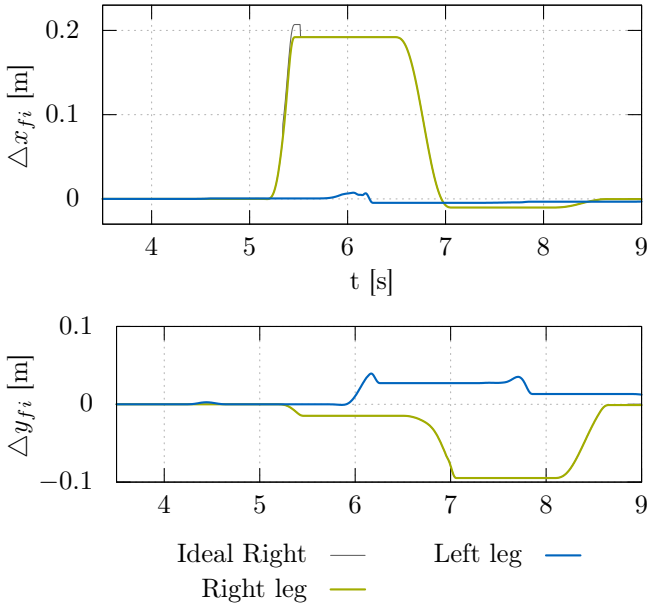


Fig. 15 Synthetic simulation example: foot trajectory adaptations for stabilizing the robot. Ideal trajectory would be without the obstacle.

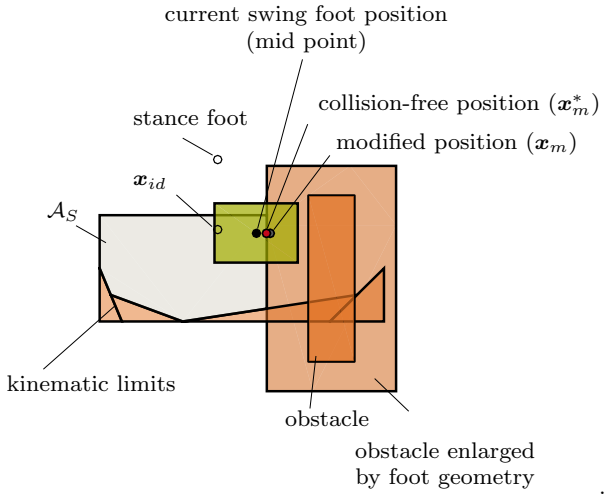


Fig. 17 Explanation of the 2D plots to visualize the overall "find safe foothold" algorithm.

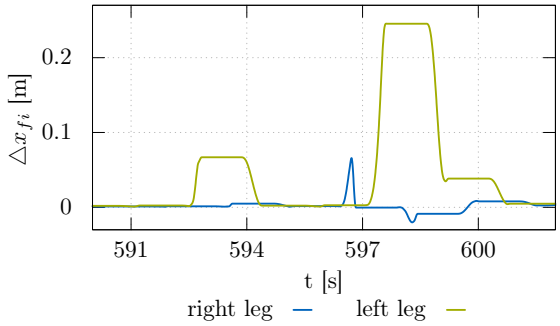


Fig. 18 Experiment 1: modification trajectories of the legs in x-direction.

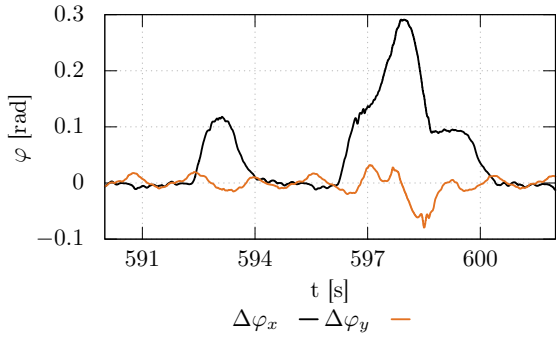


Fig. 19 Experiment 1: resulting inclination errors of the upper body (IMU data).

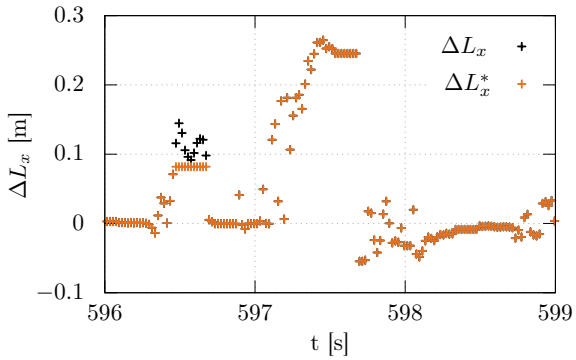


Fig. 20 Experiment 1: optimization result before (ΔL_x) and after (ΔL_x^*) collision check.

8.2 Limitations and Outlook

Although the hierarchical approach comes with many benefits, it also entails limitations. As there is no single computational instance with all information about the current setting, the method's result may not be the absolute optimum achievable. In our current implementation, the *A*-based step planner* and the *parameter optimization* evaluate the robot's next physical step. That way, step modifications caused by disturbance rejection during the current physical step cannot be taken into account during ideal motion generation. After the

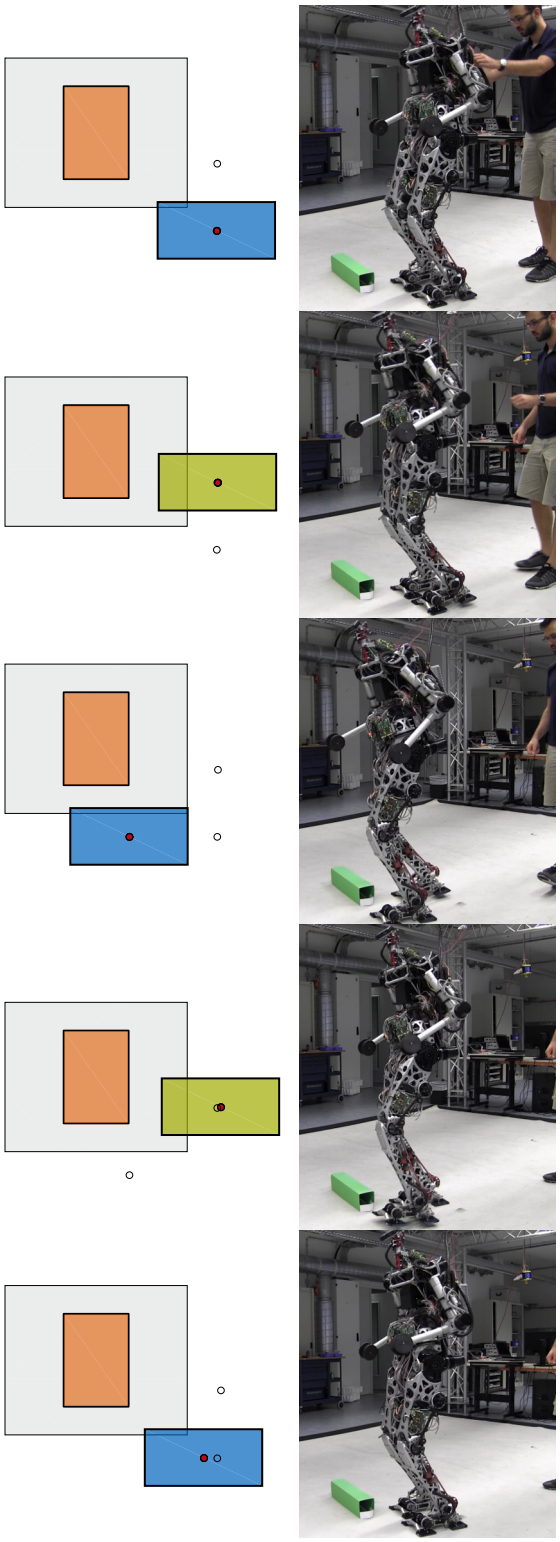


Fig. 21 Experiment 1: snapshots at different time instances - robot and polytopes.

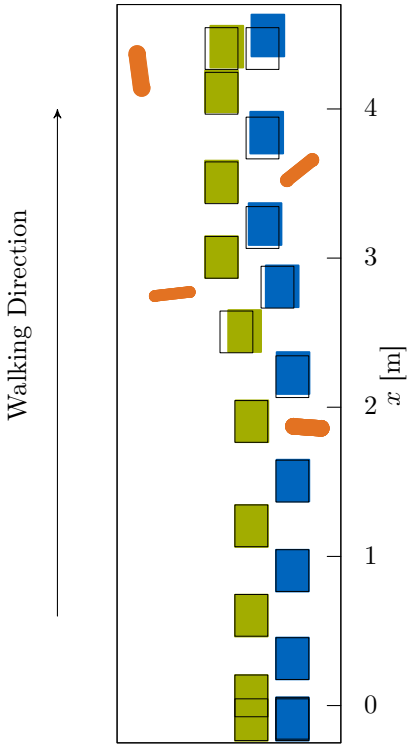


Fig. 22 Experiment 2: Ideal step sequence is represented as black boxes. Modified and executed steps are represented as filled boxes. Relevant obstacles in orange.

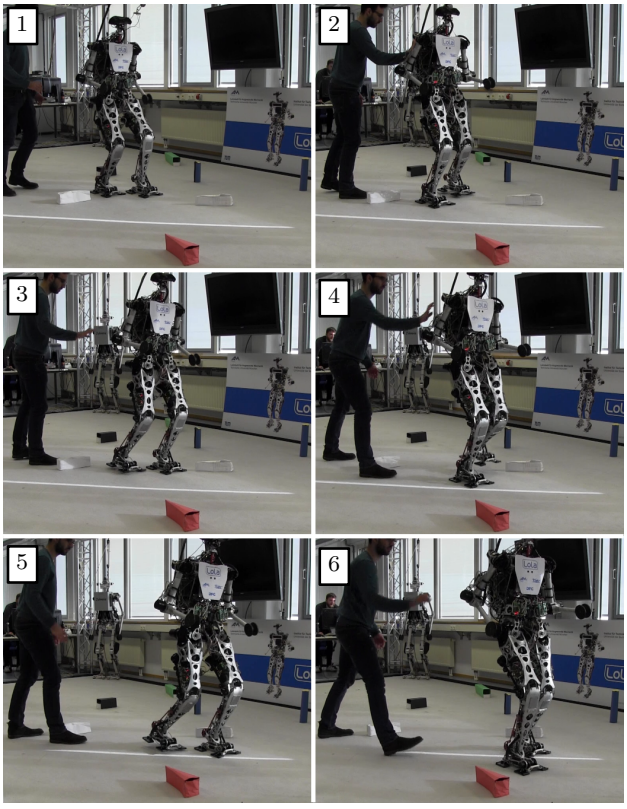


Fig. 23 Experiment 2: snapshots at different time instants.

disturbance decays, the modifications fade out and return back to the ideal walking pattern. Nonetheless, large and constant disturbances inevitably cause the real robot to diverge from the ideal planned motion. Even though the foothold positions remain collision-free, complex kinematic motions in a complex environment can always lead to possible collisions that can no longer be excluded before the step is executed. However, *Reactive Collision Avoidance* showed good performance for avoiding collisions reactively (compared to Hildebrandt et al (2014)) during step execution. This limitation of our implementation is connected with the overall question "What is more important for humanoid locomotion: Avoiding collisions or remaining stable?". In our framework, we decided to limit footstep modification for disturbance rejection by applying environmental constraints to avoid collisions. This decision is based on the assumption that disturbance rejection can take place during more than one step, whereas a collision will lead to a system failure. The limitations presented are a motivation to focus on two aspects in our future work: (1) establishing an extended feedback from the reactively working submodules to those planning the robot's ideal movement so that modified foothold positions can be taken into account during ideal planning and (2) investigating extended error handling, such as falling strategies. Whereas the first aspect (1) is mainly a question of technical implementation, the second direction of development involves hardware design modification and different control strategies. But the latter will be necessary if the biped is exposed to disturbances in real-world environments that are too severe to reject.

9 Appendix

The DAAD and the *Deutsche Forschungsgemeinschaft* (DFG project BU 2736/1-1) support this project. Special thanks go to our fantastic student Lisa Jeschek for her help in developing and implementing these ideas.

References

A Liegeois (1977) Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. In: IEEE Transactions on Systems, 12, pp 868–871

Arbulu M, Kheddar A, Yoshida E (2010) An approach of generic solution for humanoid stepping over motion. In: IEEE-RAS International Conference on Humanoid Robots, IEEE, pp 474–479

Buschmann T (2010) Simulation and Control of Biped Walking Robots. PhD thesis

Buschmann T, Lohmeier S, Ulbrich H (2010) Entwurf und Regelung des Humanoiden Laufroboters Lola. at - Automatisierungstechnik 58(11):613–621

Buschmann T, Favot V, Lohmeier S, Schwienbacher M, Ulbrich H (2011) Experiments in fast biped walking. In: IEEE International Conference on Mechatronics, pp 863–868

Chestnutt J, Takaoka Y (2010) Safe adjustment regions for legged locomotion paths. In: IEEE International Conference on Humanoid Robotics, pp 224–229

Chestnutt J, Michel P, Nishiwaki K, Kuffner J, Kagami S (2006) An intelligent joystick for biped control. In: IEEE International Conference on Robotics and Automation, IEEE, pp 860–865

Chestnutt J, Takaoka Y, Suga K, Nishiwaki K, Kuffner J, Kagami S (2009) Biped Navigation in Rough Environments Using On-board Sensing. In: IEEE/RSJ International Conference on Intelligent Robots and Systems

Deits R, Tedrake R (2014) Footstep planning on uneven terrain with mixed-integer convex optimization. In: IEEE-RAS International Conference on Humanoid Robots

Deits R, Tedrake R (2015) Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming. In: Akin HL, Amato NM, Isler V, van der Stappen AF (eds) Algorithmic Foundations of Robotics XI, Springer International Publishing, pp 109–124

Englsberger J, Ott C (2012) Integration of vertical COM motion and angular momentum in an extended Capture Point tracking controller for bipedal walking. In: IEEE-RAS International Conference on Humanoid Robots, pp 183–189

Fallon MF, Marion P, Deits R, Whelan T, Antone M, Mcdonald J, Tedrake R (2015) Continuous Humanoid Locomotion over Uneven Terrain using Stereo Fusion. In: IEEE/RAS International Conference on Humanoid Robots

Fujimoto Y, Obata S, Kawamura A (1998) Robust biped walking with active interaction control between foot and ground. In: IEEE International Conference on Robotics and Automation, vol 3, pp 2030–2035

Greene H (1983) The decomposition of polygons into convex parts. Computational Geometry 1

Guan Y, Yokoi K, Tanie K (2006) Stepping Over Obstacles with Humanoid Robots. IEEE Transactions on Robotics 22(5):958–973

Gutmann JS, Fukuchi M, Fujita M (2008) 3D Perception and Environment Map Generation for Humanoid Robot Navigation. The International Journal of Robotics Research 27(10):1117–1134

Hildebrandt AC, Wittmann R, Wahrmann D, Ewald A, Buschmann T (2014) Real-time 3D collision avoidance for biped robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 4184–4190

Hildebrandt AC, Wahrmann D, Wittmann R, Rixen D, Buschmann T (2015) Real-Time Pattern Generation Among Obstacles for Biped Robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2780–2786

Hildebrandt AC, Demmeler M, Wittmann R, Wahrmann D, Sygulla F, Rixen D, Buschmann T (2016) Real-Time Predictive Kinematic Evaluation and Optimization for Biped Robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems

Hirai K, Hirose M, Haikawa Y, Takenaka T (1998) The development of Honda humanoid robot. In: IEEE International Conference on Robotics and Automation, vol 2, pp 1321–1326

Hodgins J, Raibert M (1991) Adjusting step length for rough terrain locomotion. IEEE Transactions on Robotics and Automation 7(3):289–298

Karkowski P, Bennewitz M (2016) Real-Time Footstep Planning Using a Geometric Approach. In: IEEE International Conference on Robotics and Automation

Kuindersma S, Permenter F, Tedrake R (2014) An efficiently solvable quadratic program for stabilizing dynamic locomotion. In: IEEE International Conference on Robotics and Automation, pp 2589–2594

Kuindersma S, Deits R, Andr MF, Dai H, Permenter F, Pat K, Russ M (2015) Optimization-based Locomotion Planning, Estimation and Control Design for the Atlas Humanoid Robot. Autonomous Robots 40(3):1–27

Liu H, Liu W, Latecki LJ (2010) Convex shape decomposition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition pp 97–104

Loffler K, Gienger M, Pfeiffer F (2002) Model based control of a biped robot. In: 7th International Workshop on Advanced Motion Control, pp 443–448

Lohmeier S (2009) System design and control of anthropomorphic walking robot LOLA. IEEE/ASME Transactions on Mechatronics 14(6):658–666

Lohmeier S, Loffler K, Gienger M, Ulbrich H, Pfeiffer F (2004) Computer system and control of biped "Johnnie". In: IEEE International Conference on Robotics and Automation, IEEE, pp 4222–4227

Maier D, Lutz C, Bennewitz M (2013) Integrated perception, mapping, and footstep planning for humanoid navigation among 3D obstacles. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp 2658–2664

Michel P, Chestnutt J (2007) GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp 463–469

Naveau M, Kudruss M, Stasse O, Kirches C, Mombaur K, Souères P (2017) A ReactiveWalking Pattern Generator Based on Nonlinear Model Predictive Control. IEEE Robotics and Automation Letters 2(1):10–17

Nishiwaki K, Kagami S (2009a) Frequent walking pattern generation that uses estimated actual posture for robust walking control. In: IEEE/RAS International Conference on Humanoid Robots, pp 535–541

Nishiwaki K, Kagami S (2009b) Online Walking Control System for Humanoids with Short Cycle Pattern Generation. The International Journal of Robotics Research 28(6):729–742

Nocedal J, Wright SJ (2004) Numerical Optimization pp 1–651

Okada K, Kagami S, Inaba M, Inoue H (2001) Plane segment finder: algorithm, implementation and applications. In: IEEE International Conference on Robotics and Automation

Perrin N, Stasse O, Baudouin L, Lamiraux F, Yoshida E (2012) Fast Humanoid Robot Collision-Free Footstep Planning Using Swept Volume Approximations. IEEE Transactions on Robotics 28(2):427–439

Pratt J, Carff J, Drakunov S, Goswami A (2006) Capture Point: A Step toward Humanoid Push Recovery. In: IEEE/RAS International Conference on Humanoid Robots, pp 200–207

Rebula J, Canas F, Pratt J, Goswami A (2007) Learning Capture Points for humanoid push recovery. In: IEEE/RAS International Conference on Humanoid Robots, pp 65–72

Sabe K, Fukuchi M, Gutmann JS, Ohashi T, Kawamoto K, Yoshigahara T (2004) Obstacle avoidance and path planning for humanoid robots using stereo vision. In: IEEE International Conference on Robotics and Automation, IEEE, pp 592–597

Sarmiento A, Murrieta-Cid R, Hutchinson S (2005) A sample-based convex cover for rapidly finding an object in a 3-D environment. In: IEEE International Conference on Robotics and Automation, vol 2005, pp 3486–3491

Schwienbacher M (2012) Vertical Angular Momentum Minimization for Biped Robots with Kinematically Redundant Joints. International Congress of Theoretical and Applied Mechanics pp 8–9

Schwienbacher M, Buschmann T, Lohmeier S, Favot V, Ulbrich H (2011) Self-collision avoidance and angular momentum compensation for a biped humanoid robot. In: IEEE International Conference on

Robotics and Automation, pp 581–586

Sherikov A, Dimitrov D, Wieber Pb (2014) Whole body motion controller with long-term balance constraints. In: IEEE/RAS International Conference on Humanoid Robots, pp 444–450

Stasse O, Verrelst B, Vanderborght B, Yokoi K (2009) Strategies for Humanoid Robots to Dynamically Walk Over Large Obstacles. IEEE Transactions on Robotics 25(4):960–967

Stumpf A, Kohlbrecher S, Conner DC, von Stryk O (2014) Supervised footstep planning for humanoid robots in rough terrain tasks using a black box walking controller. In: IEEE/RAS International Conference on Humanoid Robots, pp 287–294

Tajima R, Honda D, Suga K (2009) Fast running experiments involving a humanoid robot. In: IEEE International Conference on Robotics and Automation, pp 1571–1576

Takenaka T, Matsumoto T, Yoshiike T, Hasegawa T, Shirokura S, Kaneko H, Orita A (2009) Real time motion generation and control for biped robot -4th report: Integrated balance control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 1601–1608

Urata J, Nishiwaki K, Nakanishi Y, Okada K, Kagami S, Inaba M (2011) Online decision of foot placement using singular LQ preview regulation. In: IEEE-RAS International Conference on Humanoid Robots, pp 13–18

Wahrmann D, Hildebrandt AC, Wittmann R, Sygulla F, Rixen D, Buschmann T (2016) Fast Object Approximation for Real-Time 3D Obstacle Avoidance with Biped Robots. IEEE International Conference on Advanced Intelligent Mechatronics

Wahrmann D, Hildebrandt AC, Wittmann R, Sygulla F, Seiwald P, Rixen D, Buschmann T (2017) Vision-Based 3D Modeling of Unknown Dynamic Environments for Real-Time Humanoid Navigation (in submission). International Journal of Humanoid Robotics

Whitney D (1969) Resolved Motion Rate Control of Manipulators and Human Prostheses. IEEE Transactions on Man Machine Systems 10(2):47–53

Wieber PB (2006) Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In: IEEE-RAS International Conference on Humanoid Robots, pp 137–142

Wittmann R, Hildebrandt AC, Ewald A, Buschmann T (2014) An Estimation Model for Footstep Modifications of Biped Robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2572–2578

Wittmann R, Hildebrandt AC, Wahrmann D, Buschmann T, Rixen D (2015a) State Estimation for Biped Robots Using Multibody Dynamics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2166–2172

Wittmann R, Hildebrandt AC, Wahrmann D, Rixen D, Buschmann T (2015b) Real-Time Nonlinear Model Predictive Footstep Optimization for Biped Robots. In: IEEE-RAS International Conference on Humanoid Robots, pp 711–717

Wittmann R, Hildebrandt AC, Wahrmann D, Sygulla F, Rixen D, Buschmann T (2016) Model-Based Predictive Bipedal Walking Stabilization. In: IEEE-RAS International Conference on Humanoid Robots

Yisheng Guan, Yokoi K, Tanie K (2005) Feasibility: Can Humanoid Robots Overcome Given Obstacles? In: IEEE International Conference on Robotics and Automation, IEEE, vol 1, pp 1054–1059