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

Kinematic Optimization for Bipedal Robots

A Framework for Real-Time Collision Avoidance

Arne-Christoph Hildebrandt · Simon Schwerd · Robert Wittmann · Daniel Wahrmann · Felix Sygulla · Philipp Seiwald · Daniel Rixen · Thomas Buschmann

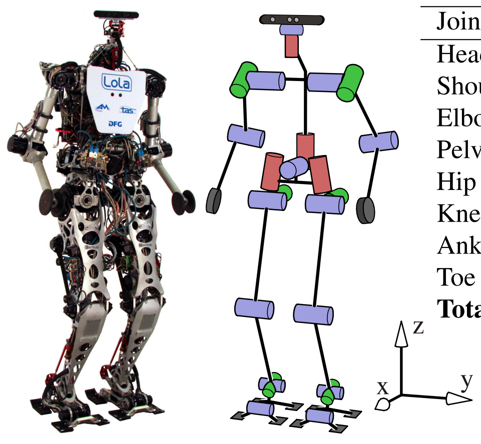
Received: date / Accepted: date

Abstract Bipedal locomotion is more than dynamically stable walking. The redundant kinematic design of humanoid robots allows for complex motions in complex scenarios. One challenge of current robotic research is the exploitation of the capacities of redundant robots in real-time applications. In this paper, we present and evaluate methods for real-time motion generation of redundant robots. The proposed methods are based on a model-predictive approach. We propose and compare methods for optimization of robot motions defined by parameterized task-space trajectories and for redundancy resolution. The approaches are successfully combined in a novel algorithm. The methods are introduced with the help of a minimal model. It shows their applicability for a wide range of complex robotic systems. We apply and validate their effectiveness and their real-time character in several experiments with different environments with the humanoid robot.*Lola*.

Keywords Bipedal Walking, Autonomous Navigation, Kinematic Optimization, Real-Time Motion Generation, Collision Avoidance

1 Introduction

One focus of current robotic research is creating robots with more autonomy to deal with the challenges of real-world environments. The robot's limited versatility seems to be one of the major bottleneck to employ them in real applications. Especially in cluttered environments, redundant hardware design is one of the possibilities to enlarge the robots' mobility. One prominent representative of redundant mobile robots are humanoids. Thanks to their redundant kinematic structure, they are predestined to be employed in diverse and complex scenarios. Motion generation for humanoid robots usually follows a hierarchical approach (Kajita et al, 2003; Nishiwaki et al, 2012; Takenaka et al, 2009; Buschmann et al, 2010; Englsberger et al, 2011): First, task space trajectories are generated to enable dynamically feasible motions based on simplified models over a long time horizon. Second, the robot's inverse kinematic is solved locally while exploiting the redundancy for secondary optimization targets. A drawback of this approach is that the robot's motion kinematics are not taken into account at the trajectory planning stage. The motion-governing workspace trajectories are calculated without feedback about the resulting kinematic motion.



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 and world coordinate system used are shown on the right side.

Arne Christoph Hildebrandt · Simon Schwerd
Technical University of Munich,
85747 Garching, GER
Phone: +49 89289 15208
E-mail: {arne.hildebrandt,simon.schwerd}@tum.de

In the worst case, this leads to workspace trajectories which are kinematically infeasible. Furthermore, kinematics are only taken into account locally. That way, the robot's redundancy, which promises a high optimization potential, cannot be exploited adequately. In this paper, we present an approach to optimize the parameterized workspace trajectories and the nullspace in a model predictive way. The parameter and the nullspace optimization are not only solved separately but also in combination. The methods are presented and analyzed using a minimal model. Furthermore, we apply them on our humanoid *Lola* (see Fig. 1). We want to emphasize, that these methods are not limited to bipedal locomotion. As shown using a minimal model, they can be applied to a much wider range of redundant robots. The paper is organized as follows: In Section 2 we give an overview of current related research projects. Section 3 discusses an optimization approach for motion planning of redundant robots. First, Section 4 provides a system overview of the experimental platform used in this work — the robot *Lola* — and the overall software framework. Furthermore, we give a detailed review of our methods for real-time motion generation. Then, the application of the presented model-predictive planning method on a humanoid robot is presented. Furthermore, the methods are analyzed in simulation and validated successfully in conducted experiments. These results are presented in Section 5. Finally, Section 6 is devoted to a conclusion and comments on future work.

2 Related Work

Most current humanoid control frameworks follow a hierarchical approach to achieve bipedal walking, which is responsive to user input or changing scenarios. These hierarchical approaches allow seperation of the navigation problem in complex environments from the walking pattern generation. Typically, the navigation problem is reduced to a search of foothold sequences to take into account geometric constraints. In bipedal locomotion, often an A*–Search is applied to search for valid step sequences on a height map (Hornung and Bennewitz, 2012; Stumpf et al, 2014; Chestnutt et al, 2009). For example, Chestnutt et al (2009) propose a navigation system based on an A*-Search on an accurate height map. The height map is set up using a laser scanner and the robot is localized with an external motion capture system. Fallon et al (2015) and Buschmann et al (2010) present two methods differing from graph-search-based approaches: Fallon et al (2015) propose to determine convex areas in a vision system into which the robot is able to step. Starting with a fixed number of steps, a mixed integer optimization is applied to calculate a step sequence. Instead of using a global map, Buschmann et al (2010) propose to analyze a set of 2D trajectories for feasibility in the vision system. The best ranked trajectory is executed by the robot. This method is very responsive to changes in the environment but does not exploit the robot's ability to step over obstacles. The navigation systems have in common that they provide foothold positions, which can be summarized as a set of parameters. This set and, additionally, user chosen parameters or parameters explicitly calculated by the step planner (Hildebrandt et al, 2015; Nishiwaki et al, 2012), as for example torso height or footstep height, describe the robot's desired motion. It separates the handling of geometric constraints imposed by the environment from the motion generation. The robot's multi-body-behavior is approximated by simple point-mass models. Using these models, reference trajectories, which are configured by the parameter set, are calculated without latencies allowing for dynamically feasible walking. The reference trajectories are set points to generate the motion on joint level at higher frequencies. Nishiwaki et al (2012); Kajita et al (2003); Takenaka et al (2009); Buschmann (2010); Englsberger et al (2011), among others, are prominent proponents of this approach for real-time motion generation. By adapting the desired motion during walking, Wittmann et al (2014); Urata et al (2011), among others, show results, taking into account sensor feedback, in rejecting unknown disturbances even in the presence of obstacles (Hildebrandt et al, 2017). Nishiwaki et al (2012); Kajita et al (2003); Takenaka et al (2009); Buschmann (2010); Englsberger et al (2011) and Wittmann et al (2014); Urata et al (2011) as well apply different methods for motion generation and different models to approximate the robot's dynamic behavior. However, they all have in common the generation of reference trajectories governing the robot's overall motion based on simplified models and heuristically chosen parameters. Buschmann et al (2005) present an approach for offline optimization of walking pattern parameters. Due to its offline character, it does not take into account the environment. Nishiwaki et al (2012) propose planning collision-free foot trajectories in the workspace connecting the desired foothold. However, the foot trajectories are calculated taking into account the kinematics of the robot only via heuristics. Baudouin et al (2011) offer an RRT-based footstep planner. They propose dividing the stepping motion into half steps and checking for collisions at predefined body configurations. The long calculation time and the sub-optimal solution provided by the RRT-algorithms makes the application in real-world environments difficult. Furthermore, the motion of the robot is defined through key-poses at half-time steps that are adapted in a post processing step to approach fluent movement.

2.1 Kinematic Planning for Bipedal Locomotion

Most of the previous approaches take the robot's kinematics into account only for a short time horizon of one control cycle. Thus, a large optimization space of the robot's redundancy is neglected. Furthermore, the robot's ability to step over or onto obstacles is only exploited by using heuristic approaches to respect kinematic limitations. Work on kinematic planning for humanoid robots that considers motion of a complete walking step focus mainly on narrow tasks such as stepping over (Guan et al, 2005, 2006; Verrelst et al, 2006; Stasse et al, 2009; Arbulu et al, 2010) or onto obstacles. Guan et al (2005, 2006) investigate the feasibility of humanoid stepping-over-motions. They propose a quasi-static trajectory planner for the task of stepping over a rectangular obstacle. Stasse et al (2009); Verrelst et al (2006); Arbulu et al (2010) take into account the Zero Moment Point (ZMP) feasibility criteria via the preview control (Kajita et al, 2003). This allows them to shift the result for stepping over an obstacle from quasi-static to dynamic robot motions. Stasse et al (2009); Verrelst et al (2006) extend the step planning process. First, they calculate the required step length and waist height to obtain a collision-free double support phase. Second, smooth swing foot trajectories are generated taking into account collision checks only in key configurations. During step execution, they propose to adapt the horizontal foot trajectory on-line. Nevertheless, to the best of our knowledge, this has not yet been validated experimentally. As opposed to Stasse et al (2009) and Verrelst et al (2006), Arbulu et al (2010) use more sophisticated body approximations for collision checks. Instead of line segments, the obstacle and the lower part of the swing leg are modeled as boxes. Arbulu et al (2010) propose, similar to Stasse et al (2009); Verrelst et al (2006), that collisions between robot and obstacle be checked only for several key configurations. Smooth swing foot trajectories are generated using clamped splines by interpolating the key configurations. The robot's full motion is generated based on the methods presented by Kajita et al (2003). The desired motions are checked for feasibility by calculating the inverse dynamics of the multibody model. The presented methods allow humanoid robots to step over one large obstacle at a time. However, they have the following disadvantages:

– They all use very simplified geometric models, so complex 3D geometries of the robot or the environment cannot be represented adequately.
– They only consider collisions between the lower legs and one obstacle. Neither potential self-collisions, nor simultaneous collisions with several obstacles are taken into account.
– They limit the foot movements to a plane. More general movements which exploit all the robot's DoFs are not considered.
– The methods are presented as stand-alone: They are not integrated in frameworks involving perception and navigation. Thus, their compatibility with a whole motion-generation framework has to be proven.
– They concentrate on the stepping motion over one obstacle. The robot's whole kinematic is not optimized and it is not considered in more general walking.

The method presented by Koch et al (2014) is a more general approach. They generate the stepping-over-motion from a whole-body-motion-optimization. The method is applied to the test case of a stepping-over motion over one obstacle. However, it is an off-line method and not applicable to real-time applications. In contrast to these work, Nishiwaki (2011) propose a method to design torso trajectories to take into account future kinematic limits, but the environment is neglected.

2.2 Motion Planning for Redundant Robots

Considering the stepping motion of humanoid robots in cluttered environments from a point of view of whole-body-motion optimization seems important. Whole-body-motion optimization allows for a more complete and more general exploitation of the robot's capabilities, not limiting the applicability of the algorithm to specific tasks. In recent years, many frameworks for motion planning were developed (Zucker et al, 2013; Schulman et al, 2013; Chitta et al, 2012). In their objective to develop a motion planning framework for any kind of robotic system lies the difficulty to apply it to our application: they neither exploit nor take into account the characteristics of humanoid walking. This makes it difficult to satisfy the hard constraints of dynamic humanoid walking. Motion planning for humanoid robots has to satisfy hard real-time constraints, it has to take the dynamics of bipedal walking into account and it has to react to unknown perturbations. Zucker et al (2013); Schulman et al (2013), among others, present powerful gradient-based frameworks: Zucker et al (2013) introduces the algorithm *CHOMP*. It minimizes a cost function by covariant gradient descent taking into account the path smoothness and penalizing collisions. Distance gradients are calculated based on pre-computed distance fields to accelerate the computations. Schulman et al (2013) name the presented algorithm *TRAJOPT*. It is an approach based on sequential convex optimization, which takes the system's boundary conditions into account as cost functions. Both algorithms have been applied to legged locomotion: Schulman et al (2013) present results for planning foot placements while maintaining static stability under environmental constraints for the humanoid *ATLAS*. Zucker et al (2013) apply *CHOMP* on the four-legged robot *LittleDog* walking on uneven terrain. Gienger et al (2008) followed an approach similar to the one presented in this work. Trajectories were found using linear attractor dynamics with control points. The control points were optimized by a gradient-based optimization algorithm. Depending on the number of control points and the search space, the computation time could be adapted depending on the problem. Since the methods were developed for grasping motions, they do no respect the hard timing and geometric constraints of humanoid locomotion, e.g. foot-ground contact. This is critical in order to maintain balance. An optimization method, which is similar to our approach introduced in Schuetz et al (2014) and used in Subsection 3.5, was presented by Tassa et al (2012). The presented extensions to the *Differential Dynamic Programming* approach allows for almost real-time use. Schuetz (2017) discusses the differences and similarities between Schuetz et al (2014) and Tassa et al (2012).

3 Predictive Kinematics

In the following, the proposed methods are introduced with an illustrative example. We review and compare our methods presented in Hildebrandt et al (2016) and Schuetz et al (2014). While we propose in Hildebrandt et al (2016) a parameter optimization to optimize the motion of a bipedal robot, we present in Schuetz et al (2014) a model-predictive approach to exploit the redundancy of an agricultural manipulator. The illustrative example emphasizes the common characteristics of kinematic optimization of redundant robots and our method's applicability to a wide range of problems. Furthermore, we combine parameter optimization and continuous nullspace optimization for motion planning of redundant robots.

3.1 Problem Statement

A task often posed for serial robots is the motion of their end-effector from a point A to a point B in space. One way to generate the robot's motion is to describe the path of the end-effector from A to B via trajectory primitives in workspace. The trajectories $\boldsymbol{w} = \boldsymbol{w}(\boldsymbol{p})$ are configurable with parameters $\boldsymbol{p}$, which can be adapted with respect to the overall desired motion. For redundant robots the configuration space's dimension n is larger than the workspace dimension m $(n > m)$. This kinematic characteristic needs to be resolved during motion generation. Common approaches such as the *Resolved Motion Rate Control* (Whitney (1969)) or *Automatic Supervisory Control* (A. Liegeois (1977)) can be used to solve the redundancy on velocity level at each time step t_i. Using these approaches, the inverse kinematics results in

$$\dot{\boldsymbol{q}}(t_{i+1}) = \boldsymbol{f}(\boldsymbol{q}(t_i), \boldsymbol{u}(t_i), \boldsymbol{p}, t_i) \tag{1}$$

$$= \boldsymbol{J}_w^{\#} \dot{\boldsymbol{w}}(\boldsymbol{p}, t_i) - \alpha \boldsymbol{N} \boldsymbol{u}(t_i) \tag{2}$$

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_0 \quad t \in [t_0, t_{end}] \tag{3}$$

with the Moore-Penrose-Inverse of the Jacobian matrix $\boldsymbol{J}_w^{\#}$ of $\dot{\boldsymbol{q}}$ with respect to $\dot{\boldsymbol{w}}$ and the nullspace projection matrix $\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{J}_w^{\#} \boldsymbol{J}_w$. For the sake of simplicity, we omit the time dependency in the following. The vectors $\boldsymbol{q}, \dot{\boldsymbol{q}}$ denote the joint angles resp. angular velocities of the robot. α is a weighting factor and $\boldsymbol{u}$ is the nullspace input which can be freely chosen. The advantage of this approach compared to a complete generation of motion directly in configuration space is a reduction of complexity. First, the parameterized motion of an end-effector can be generated in a more simple way in task space and second, redundancy can be exploited for optimality without interfering with the end-effector motion. That way, the computational complexity can also be reduced, which makes this approach applicable in real-time application such as, for example, dynamic bipedal walking of humanoids with their high number of DOF design or redundant serial manipulators. In the following, we discuss several methods to determine the free variables $\boldsymbol{u}$ and $\boldsymbol{p}$.

3.2 Application Scenario

We introduce a 5-DOF manipulator, performing a predefined planar motion of two straight trajectories which connect three points $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$ in an environment with two obstacles (see Fig. 2). $\boldsymbol{w}_2$ can be understood as a path point whose horizontal coordinate can be configured by a parameter p. In this case, the desired workspace trajectory $\boldsymbol{w}_d(t)$, defined by $\boldsymbol{w}_1$, $\boldsymbol{w}_2(p)$ and $\boldsymbol{w}_3$, is supposed to be collision free, but the robot's links may collide with the obstacle. The timing of $\boldsymbol{w}_d(t)$ is defined via the fixed time points at $\boldsymbol{w}_1$, $\boldsymbol{w}_2(p)$ and $\boldsymbol{w}_3$.

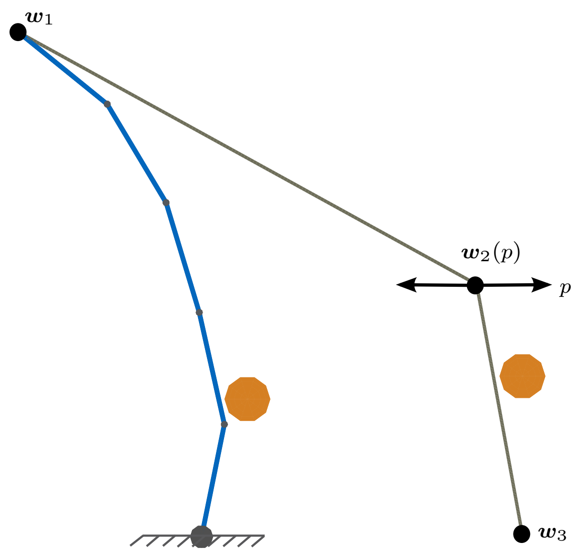


Fig. 2: Model: redundant robot in blue, parameterized workspace trajectory in grey, obstacles in orange.

3.3 Local Kinematics

A. Liegeois (1977) proposed to choose the input $\boldsymbol{u}$ as the gradient of a cost function to exploit the robot's redundancy. The inverse kinematics of (1) lead to

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}) = \boldsymbol{J}_w^{\#}\dot{\boldsymbol{w}} - \alpha \boldsymbol{N}(\frac{\partial L}{\partial \boldsymbol{q}})^T \tag{4}$$

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_0, \quad t \in [t_0, t_{end}]\,. \tag{5}$$

The vector $\frac{\partial L}{\partial \boldsymbol{q}}$ is the gradient of a cost function, which we defined as

$$\begin{aligned} L =& c_{coll,self} L_{coll,self} + c_{coll,dist} L_{coll,dist} + \dots \\ &+ c_{vel} L_{vel} + c_{cmf} L_{cmf}\,. \end{aligned} \tag{6}$$

L is the sum of cost functions which penalizes collisions of the robot links with themselves ($c_{coll,self} L_{coll,self}$) and obstacles ($c_{coll,dist} L_{coll,dist}$) as well as high angular velocities ($c_{vel} L_{vel}$). The cost function $c_{cmf} L_{cmf}$ is used to define a comfort pose (Schwienbacher et al, 2011). This formulation also applies to the inverse kinematic module of the walking control system of humanoid robots. In the following, we will refer to (4) and (5) as *local control.*

3.4 Taskspace Optimization

In many robotic applications, it is not important that the end-effector exactly follows a pre-defined workspace trajectory, but that the end-effector moves from A to

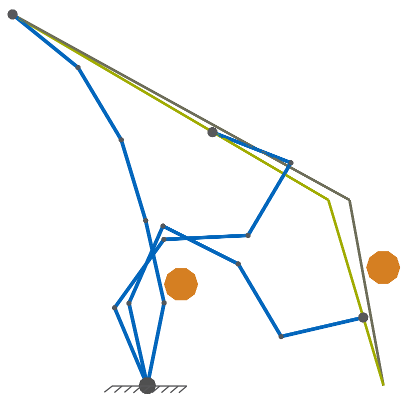


Fig. 3: Task space trajectory before (gray) and after parameter optimization (green)

B. In Hildebrandt et al (2016) we propose a trajectory representation based on splines which are configurable using parameters. Since the optimization space consists of only a few parameters the workspace trajectory describing a bipedal stepping motion can be optimized in real-time taking into account the robot's whole kinematic model. This method is applied as follows to our model introduced in Subsection 3.2: we define the horizontal coordinate of the second point $\boldsymbol{w}_{2,x}(p)$ to be dependent on a single parameter p (see Fig. 2). Note that any parametric movement representation $\boldsymbol{w}(t, p)$ could be applied to the presented method, but we consider this to be one of the most illustrative ones. The optimization problem is formulated as follows

$$\min_p \phi(\boldsymbol{q}) = g(\boldsymbol{q})|_{t_e} + \int_{t_0}^{t_e} L(\boldsymbol{q})dt \tag{7}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, p) = \boldsymbol{J}_w^{\#}\dot{\boldsymbol{w}}(p) - \alpha \boldsymbol{N}\frac{\partial L}{\partial \boldsymbol{q}} \tag{8}$$

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_0 \tag{9}$$

with $g(\boldsymbol{q})|_{t_e}$ denoting a cost term at $t = t_e$. As described in Hildebrandt et al (2016) the optimization problem can be solved using a gradient method for the parameter p. Fig. 3 shows three snapshots of the resulting task space trajectory. p is optimized such that it is shifted away from the obstacle to avoid collision costs.

3.5 Redundancy Exploitation

In Schuetz et al (2014), we presented a model-predictive approach to solve the inverse kinematics and to exploit the nullspace of redundant manipulators. Instead of using a discrete parameter as optimization variable, we search for a continuous input $\boldsymbol{u}_{opt}$ as proposed in Nakamura and Hanafusa (1987). It replaces the local-acting

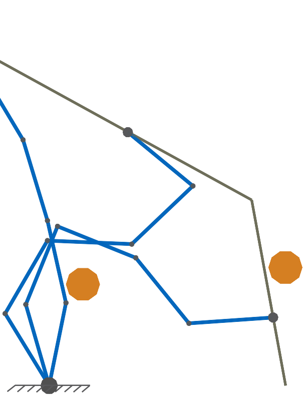


Fig. 4: Motion with global exploitation of redundancy

gradient $\frac{\partial L}{\partial \boldsymbol{q}}$. We get

$$\min_{\boldsymbol{u}} \phi(\boldsymbol{q},\boldsymbol{u}) = g(\boldsymbol{q},\boldsymbol{u})|_{t_e} + \int_{t_0}^{t_e} L(\boldsymbol{q},\boldsymbol{u})dt \tag{10}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q},\boldsymbol{u},p) = \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}(p) + \alpha \boldsymbol{N}\boldsymbol{u} \tag{11}$$

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_0 \tag{12}$$

As proposed in Schuetz et al (2014) the optimization problem can be solved applying *Pontryagin's Minimum Principle*. It leads to an unconstrained optimization problem for $\boldsymbol{u}$ which is solvable with gradient methods. Furthermore, we extended the formulation from velocity to acceleration level in Schuetz et al (2014) avoiding discontinuous angular velocities. Fig. 4 illustrates the resulting motion. Treating the nullspace control as a global optimization problem leads to a more predictive motion, which is superior in avoiding the obstacle located near the kinematic links of the robot.

3.6 Taskspace Optimization & Redundancy Exploitation

In Subsection 3.4 and Subsection 3.5, we presented methods for parameter optimization of workspace trajectories and for optimization of a continuous input vector to exploit the robot's nullspace. Having a parameterized workspace trajectory and the continuous input vector $\boldsymbol{u}$ the combined optimization problem results in

$$\min_{\boldsymbol{u},\ p} \phi(\boldsymbol{q},\boldsymbol{u}) = g(\boldsymbol{q},\boldsymbol{u})|_{t_e} + \int_{t_0}^{t_e} L(\boldsymbol{q},\boldsymbol{u})dt \tag{13}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q},\boldsymbol{u},p) = \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}(p) + \alpha \boldsymbol{N}\boldsymbol{u} \tag{14}$$

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_0 \tag{15}$$

Bocek (1980) presented an optimization formulation to simultaneously optimize a discrete parameter describing the system and a continuous system input. We successfully applied the method in Wittmann et al (2016) for optimizing foothold positions and continuous CoM motions to stabilize a bipedal robot under external disturbances. We apply this formulation as follows to combine the previously presented methods: by approaching the computation of the inverse kinematics as the combined optimization problem stated in equation (13) - (15), we define the Hamilton function as:

$$H(\boldsymbol{q},\boldsymbol{\lambda},\boldsymbol{u},p) = L(\boldsymbol{q},\boldsymbol{u}) + \boldsymbol{\lambda}^T \boldsymbol{f}(\boldsymbol{q},\boldsymbol{u},p) \tag{16}$$

with the Lagrange multiplier $\boldsymbol{\lambda}$. In the application of the optimal kinematic planning approach, the nullspace control vector as well as the task space parametrization are constrained variables. Therefore, we apply *Pontryagin's Minimum Principle* to the combined optimization problem. We obtain the following optimization conditions:

$$\dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}(\boldsymbol{q},\boldsymbol{u}) \tag{17}$$

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{q}} = -\frac{\partial L}{\partial \boldsymbol{q}} - \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\lambda} \tag{18}$$

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_0 \tag{19}$$

$$\boldsymbol{\lambda}(t_{end}) = 0 \tag{20}$$

$$\boldsymbol{u}_{opt} = \arg\min_{\boldsymbol{u}} H(\boldsymbol{q}_{opt},\boldsymbol{\lambda}_{opt},\boldsymbol{u},p_{opt}) \tag{21}$$

$$\left[\left(\frac{\partial g}{\partial p}\right)_{t_e} + \int_{t_0}^{t_e} \frac{\partial H}{\partial p} dt\right]^T \delta p \geq 0 \tag{22}$$

The boundary condition (19) comes from the given initial position of the robot and those in (20) stem from its free final form regarding the nullspace and the parameter. With the conditions (17) - (22) we calculate $\boldsymbol{\lambda}$ and $\boldsymbol{q}$ for any given nullspace control $\boldsymbol{u}$ and parameter p. This allows us to solve the problem by applying a projected conjugate gradient algorithm.

3.7 Conjugate Gradient Method

It is well known that the descent gradient algorithm shows slow convergence, therefore we apply a nonlinear conjugate gradient method, described in Bocek (1980), for the constrained nonlinear optimal control with parametrization. This algorithm provides a computationally efficient solution of the optimal control problem. Furthermore, it has the advantage for real-time applications that the optimization can be aborted at any given time. Thus, the robot can execute an intermediate, but current best, solution. To speed up convergence, we can also use the solution of $\boldsymbol{u}(t)$ calculated by the *local control* as a good initial solution $\boldsymbol{u}_0$ for the algorithm. The basic algorithm integrates the differential equation of the inverse kinematics (14) forward and the adjoint variable (18) backward in time. The input $\boldsymbol{u}(t)$ and the parameter $\boldsymbol{p}$ will be modified according to the gradients with a stepsize β. The correction stepsize may be chosen as either fixed or adaptable by linesearch algorithms. A detailed description is shown in Algorithm 1.

Algorithm 1 Conjugate Gradient method for combined optimization

```
$i \leftarrow 0$
$p^0 \leftarrow p_{initial}$
Converged $\leftarrow$ false
$\boldsymbol{q}^0(t) \leftarrow \int_{t_0}^{t_e} \dot{\boldsymbol{q}}(\boldsymbol{u}_0(t), p^0)\mathrm{d}t$ (using *local control* and (19))
$J^0 \leftarrow \int_{t_0}^{t_e} L(\boldsymbol{q}^0(t), \boldsymbol{u}^0(t))\mathrm{d}t$
**while** Converged $\neq$ true **do**
    $\boldsymbol{\lambda}^i(t) \leftarrow \int_{t_e}^{t_0} \dot{\boldsymbol{\lambda}}(\boldsymbol{q}^i(t), \boldsymbol{u}^i(t), p^i)\mathrm{d}t$
(from (18) with (20), backward in time: $t_e \rightarrow t_0$)
    $\boldsymbol{g}_u^i \leftarrow \frac{\partial H}{\partial \boldsymbol{u}}^i$
    $g_p^i \leftarrow \frac{\partial g}{\partial p}|_{t_e}^i + \int_{t_0}^{t_e} \frac{\partial H}{\partial p}^i \mathrm{d}t$
    **if** $i \neq 0$ **then**
        $\beta^i \leftarrow \frac{\int_{t_0}^{t_e} \boldsymbol{g}_u^i \boldsymbol{g}_u^i \mathrm{d}t}{\int_{t_0}^{t_e} \boldsymbol{g}_u^{i-1} \boldsymbol{g}_u^{i-1} \mathrm{d}t}$
        $\gamma \leftarrow \frac{\boldsymbol{g}_p^i \boldsymbol{g}_p^i}{\boldsymbol{g}_p^{i-1} \boldsymbol{g}_p^{i-1}}$
        $\boldsymbol{s}^i \leftarrow -\boldsymbol{g}_u^i + \beta^i \boldsymbol{s}^{i-1}$
        $c^i \leftarrow -g_p^i + \gamma^i c^{i-1}$
    **else**
        $\boldsymbol{s}^i \leftarrow -\boldsymbol{g}_u^i$
        $c^i \leftarrow -g_p^i$
    **end if**
    $\boldsymbol{u}^{i+1}(t) \leftarrow \boldsymbol{u}^i(t) + \alpha^i \boldsymbol{s}^i(t)$
    $p^{i+1} \leftarrow p^i + \alpha^i c^i$
    $i \leftarrow i + 1$
    $\boldsymbol{q}^i(t) \leftarrow \int_{t_0}^{t_e} \dot{\boldsymbol{q}}(\boldsymbol{u}^i(t), p^i)\mathrm{d}t$
    $J^i \leftarrow \int_{t_0}^{t_e} L(\boldsymbol{q}^i(t), \boldsymbol{u}^i(t))\mathrm{d}t$
    Converged $\leftarrow |\frac{J^i - J^{i-1}}{J^{i-1}}| < tol$
**end while**
```

3.8 Results

We analyzed the presented methods in a test case with the model presented before and two obstacles. The workspace trajectories are collision-free. Fig. 5 shows snapshots of the resulting motion using the *local control* and the combined optimization method. The influence of the methods on the parameter and the nullspace is clearly visible. We can observe that the local character of the *local control* leads to a disadvantageous situation during the second half of the motion. It almost leads to a collision. Contrary to that, the global optimization of both, task space parameter and nullspace control, shows a more *predictive* character as expected. Fig. 6 shows the resulting costs over time for all four methods. Tab. 1 shows the resultant parameter and values of the cost functions. Although parameter and nullspace optimization show separately impressive results, the combined optimization is able to further reduce the costs.

Table 1: Detailed results of optimization process showing integrated cost parts over time. For total cost over time, see Fig. 6. Reference parameter $p = 60.0$

	p	L_{vel}	L_{cmf}	$L_{coll,self}$	$L_{coll,obs}$
loc	const.	6.1	9.0	84.7	146.24
par	53.72	6.3	9.3	85.4	136.88
nul	const.	6.5	9.4	84.5	126.27
com	37.49	14.2	12.9	85.2	42.38

4 Application to Humanoids

This section presents the application of the methods introduced in Section 3 to the humanoid robot *Lola*: first, we give a hardware and software overview. Then, we present in Subsection 4.2 the kinematic planing in more detail and show the similarities to the model used in Section 3. Finally, we integrate the kinematic planing in our current control architecture.

4.1 System Overview

4.1.1 Hardware Overview

Our humanoid robot *Lola* (see Fig. 1) weighs approximately 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 7 DoFs and of the pelvis with 2 DoFs. For the robot's stabilization, we use an Inertial Measurement Unit (IMU) and two 6-axis force-torque sensors (FTS). The IMU is located in the upper body and the FTS in both ankles. For environment perception, we set up an Asus Xtion PRO LIVE RGB-D camera[1]. It is mounted on the robot's head and can be actuated by a pan/tilt unit. The robot is equipped with two on-board computers. One is used for vision processing

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

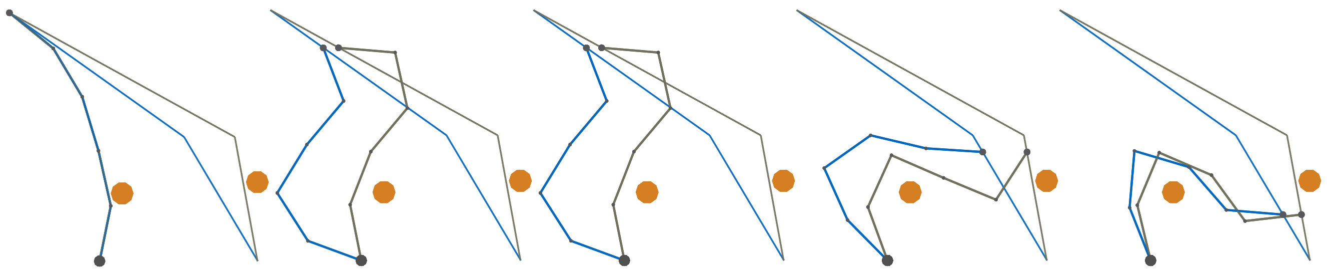


Fig. 5: Comparison of initial motion calculated with the *local control* (grey) and motion with combined optimization (blue)

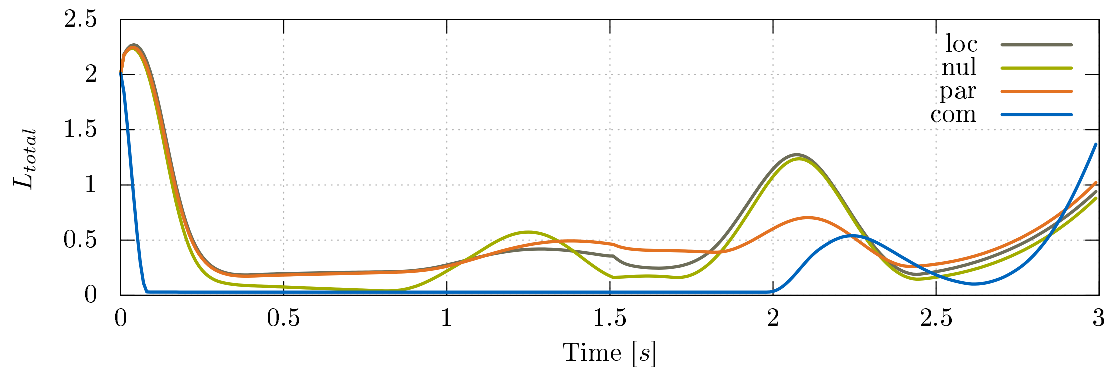


Fig. 6: Total cost L over time for local (*loc*), nullspace (*nul*), parameter (*par*) and combined optimization (*com*).

and the other is used for walking control. The computer used for vision processing runs under a Linux OS and the other under a QNX-RTOS. Both have an Intel Core i7-4770S@3.1GHz (4x) processor and 8GB RAM. They comunicate via Ethernet using TCP. Lohmeier et al (2009); Buschmann et al (2011) give more detailed informations.

4.1.2 Control Overview

An overview of the control architecture is depicted in Fig. 7. The *vision system* (Wahrmann et al, 2016)[2] approximates the environment in our collision world representation as 3D *swept-sphere-volumes* (SSV-objects) or surfaces the robot is able to step onto. The environment representation is used in the *global control* and the *feedback control* for collision avoidance. Before each step, the *global control* generates the ideal walking pattern for the next n_{Steps} steps. The walking pattern includes the center of pressure (CoP) reference trajectories and the ideal workspace trajectories $\boldsymbol{w}_{id}(t) \in \mathbb{R}^m$. The ideal workspace trajectories are composed of the center of mass (CoM) position, torso rotations, and position and orientation of the feet. The robot's walking pattern is configurable by a parameter set $\boldsymbol{p}_{wp}$, which contains the robot's foothold for each step, parameters describing the robot's movements (e.g. CoM or swing-foot height) and the step time T_{Step}. Based on user input and the perceived environment the *A*-based step planner & parameter optimization* calculates a sequence of steps and determines T_{step}. Furthermore, it evaluates and optimizes $\boldsymbol{p}_{wp}$ using a full kinematic model. Subsection 4.2 gives more details. The *walking pattern generation* uses a simplified multi-body model and $\boldsymbol{p}_{wp}$ to calculate $\boldsymbol{w}_{id}(t)$. The simplified multi-body model is a three-mass-model to account for dynamic effects caused by fast leg movements. This is especially important during fast-walking as well as when dynamically walking in cluttered environments[3]. Details are presented in Buschmann et al (2010). During step execution, the methods presented in Wittmann et al (2016) reject external perturbations by modifying the desired foot trajectories. The *feedback control* uses $\boldsymbol{w}_{id}(t)$ as set points to calculate the desired joint target data $\boldsymbol{q}_d \in \mathbb{R}^n$ and $\dot{\boldsymbol{q}}_d \in \mathbb{R}^n$. The desired motion at time t_k, $\boldsymbol{w}_{id,k} = \boldsymbol{w}_{id}(t_k)$ is modified locally to take sensor input into account. The *reactive collision avoidance* optimizes $\boldsymbol{w}_{id,k}$ to reactively prevent collisions with obstacles and self-collisions. The *walking stabilization* modifies $\boldsymbol{w}_{id,k}$

[2] Available under `https://github.com/am-lola/lepp3`.

[3] Compare the videos of the experiments in `https://youtu.be/6diLLVv41Vw` or in `https://youtu.be/rKsx8HKvBkg`.

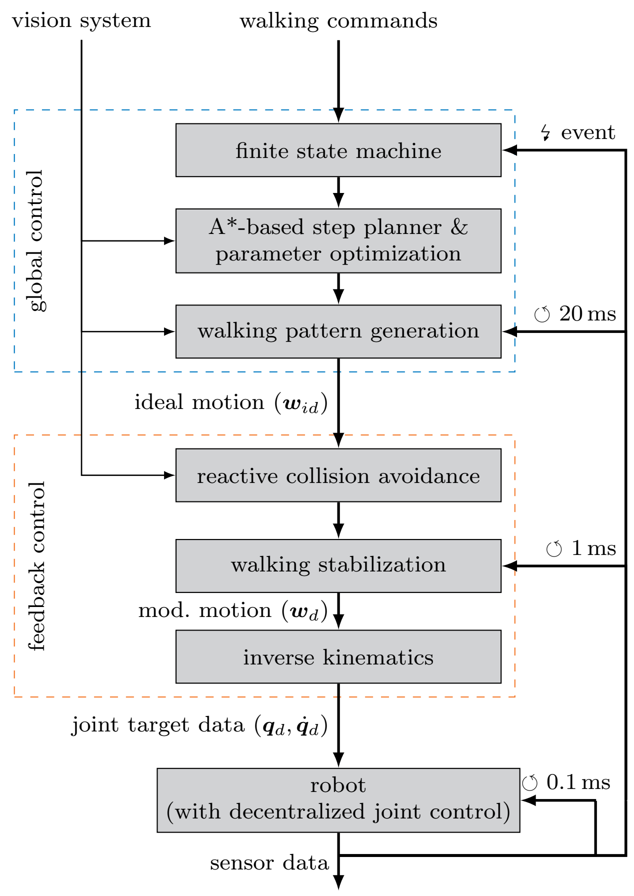

vision system
walking commands
global control
finite state machine
↯ event
A*-based step planner & parameter optimization
walking pattern generation
20 ms
ideal motion ($\boldsymbol{w}_{id}$)
feedback control
reactive collision avoidance
walking stabilization
1 ms
mod. motion ($\boldsymbol{w}_d$)
inverse kinematics
joint target data ($\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d$)
robot (with decentralized joint control)
0.1 ms
sensor data

Fig. 7: *Lola*'s real-time walking control system.

to stabilize the robot. More details can be found in Buschmann et al (2011). The modified workspace trajectories $\boldsymbol{w}_{d,k}$ resp. $\dot{\boldsymbol{w}}_{d,k}$ are input to the *inverse kinematics* on velocity-level (Whitney, 1969; A. Liegeois, 1977), to solve for the joint space velocities $\dot{\boldsymbol{q}}_{d,k} \in \mathbb{R}^n$ from $\dot{\boldsymbol{w}}_{d,k}$. The robot's redundancies ($m < n$) are exploited at each control cycle to minimize a cost function L_y. The motion in the nullspace of the robot is used for minimization of angular momentum considering constraints as self-collision avoidance and joint limits (Schwienbacher et al, 2011; Schwienbacher, 2012). The calculated $\boldsymbol{q}_{d,k}, \dot{\boldsymbol{q}}_{d,k}$ are then processed by the distributed joint controllers. The decentralized concept of the joint controller allows for high sampling rates (50 µs current, 100 µs velocity and position) for the cascaded feedback loops. This way joint tracking errors of less than 2 mrad can be achieved in all joints.

4.2 Model-Predictive Kinematic Planning

In the hierarchical approach as described in Subsection 4.1, once before each walking step of the robot the workspace trajectories $\boldsymbol{w}_{id}(t)$ are planned. First, foot trajectories and the vertical CoM trajectory are planned. These trajectories consist of splines as described in Hildebrandt et al (2016). They depend fully on the parameter $\boldsymbol{p}_{wp}$. The foot trajectories, the vertical CoM trajectory and the desired CoP trajectory are the input to the method presented in (Buschmann et al, 2007). The method uses a three-mass-model to calculate the horizontal CoM movement over a time horizon of three steps to allow for dynamically feasible bipedal walking. Thus, the robot's kinematics are described in workspace by:

– the trajectories of six DoFs for each foot;
– two trajectories for toe rotations;
– three trajectories describing the CoM movement;
– an upright torso (imposed in our work).

At each control cycle t_k the robot's joint speeds are calculated by combining the nullspace optimization with an additional gradient descent in a subspace of the task space $\boldsymbol{w}$:

$$\dot{q}_k = J_w^{\#}\dot{\boldsymbol{w}}_k - (I - J_w^{\#}J_w)\boldsymbol{u} \quad (23)$$

$$S\dot{\boldsymbol{w}}_k = S\dot{\boldsymbol{w}}_{id,k} - \dot{\boldsymbol{x}}_{RCA} \quad (24)$$

with the modification term for collision avoidance and the local inputs $\boldsymbol{u}$

$$\dot{\boldsymbol{x}}_{RCA} = [(S^* J_w)^{\#}]^T \nabla_{\dot{\boldsymbol{q}}} L_{RCA}, \quad (25)$$

$$\boldsymbol{u} = \nabla_{\dot{\boldsymbol{q}}} L_y. \quad (26)$$

$J_w, J_w^{\#}$ are the Jacobian of $\dot{\boldsymbol{w}}_k$ with respect to $\dot{\boldsymbol{q}}_k$ and its pseudoinverse. $\nabla_{\dot{\boldsymbol{q}}} L_y$ is the gradient to L_y (cf. Subsection 4.1.2) and $\nabla_{\dot{\boldsymbol{q}}} L_{RCA}$ is a gradient devoted to avoid collisions and joint limits. S, S^* are selection matrices for the relative and absolute swing foot coordinates. More detailed informations can be found in Hildebrandt et al (2014). This hierachical approach allows for real-time bipedal walking control. A drawback of the approach is that the full kinematic movement of the robot is not considered before execution of each step. Previously, $\boldsymbol{p}_{wp}$, which configure the workspace trajectories, were set heuristically without predictive feedback about the kinematic movement. Since also the *reactive collision avoidance* is a local method, complex motions in cluttered environments could be kinematically not feasible. The problem as described for our framework is true for most of the current frameworks for real-time control of bipedal walking. Kajita et al (2003); Englsberger et al (2011); Nishiwaki et al (2012), among others, are using simple models to allow for dynamically feasible movement. They do not take the kinematic motion over a long time horizon into account. For this reason, we proposed in Hildebrandt et al (2016) to extend the *global control* by a model-predictive kinematic evaluation and optimization. In the following, we review our model predictive approach for parameter evaluation and optimization. Furthermore, we present the integration of the methods presented in Section 3.

4.2.1 Model

Our model-predictive approach uses the robot's kinematic model as depicted in Fig. 1. It takes the *feedback control* without external influences or sensor feedback into account. The equations describing the robot's kinematics (23)-(25) can be summarized as a first order differential equation of the form

$$\begin{pmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{w}} \end{pmatrix} = f(\boldsymbol{q}, \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{w}_{id}, \dot{\boldsymbol{w}}_{id}). \tag{27}$$

Due to real-time constraints, we use a time horizon of one physical step of the robot for time integration. The method could be extended to take multiple steps into account just by integrating over a longer time horizon. The initial conditions are determined by the state of the model at the end of the previous step. The differences to Section 3 are justified by the different kinematic structure and the dynamic constraints of bipedal locomotion: $\boldsymbol{x}_{RCA}$ adapts $\boldsymbol{w}_{id}$ locally. It depends only on the robot's direct kinematics. The local adaption of the ideal planned foot trajectories is necessary due to the geometric extension of the robot's end effector, its foot, and the limited number of control points. Without local adaptation, collision-free end effector movements would be only realizable using a high number of control points. The resulting search space could not be handled in the limited calculation time. Additionally, $\boldsymbol{x}_{RCA}$ is used to react to local changes during execution of the ideal planned motion in real-world environments. Contrary to the model described in Section 3, workspace trajectories are also used to meet the dynamic constraints of bipedal locomotion. Thus, the trajectories describing the horizontal CoM movements are determined by the method presented in Buschmann et al (2007) and cannot be changed without influencing the stability.

4.2.2 Parameter Set

The robot's kinematic motion is governed by $\boldsymbol{w}_{id}$ as explained in Subsection 4.2. The trajectories are splines which are fully described by $\boldsymbol{p}_{wp}$ except for the horizontal CoM trajectories which are calculated to meet dynamic constraints. In this work, we focus on the subset $\boldsymbol{p}_{opt}$ of $\boldsymbol{p}_{wp}$ for online application. $\boldsymbol{p}_{opt}$ consist of the parameter h_{CoM} and the set of control points $\boldsymbol{p}_{SF}$ describing the height of the CoM resp. the swing-foot's sagital and lateral movement. The CoM height is determined via piecewise 5th order polynomials connecting start and finishing height of each step. The finishing height is chosen by h_{CoM}. In Hildebrandt et al (2016), we introduced a trajectory representation for the swing-foot's sagital and lateral movement. It is based on cubic splines connecting supporting points. This representation allows for smooth foot-ground touchdown on acceleration level, which is crucial for stable walking. The shape of the trajectory can be determined based on $\boldsymbol{p}_{SF}$ representing the supporting point positions. The dimension of $\boldsymbol{p}_{SF}$ is not fixed, but can be chosen depending on the problem.

4.2.3 Optimization

The methods introduced in Section 3 on the example of a 5-DOF manipulator can be applied directly to the prediction model (see Subsection 4.2.1). It represents a redundant robot with $m < n$ as explained in Subsection 4.1.2. The system's equation depends on workspace trajectories ($\boldsymbol{w}_{id}$), which are fully described by a set of parameters ($\boldsymbol{p}_{wp}$) and which are calculated before each step. The redundancy of the system is exploited to minimize a cost function. The input vector $\boldsymbol{u}$ governing the nullspace motion is a continous input to the system. The input $\boldsymbol{u}$ is currently calculated for each control cycle step using the *local control* as the gradient to L_y. Applying the methods introduced in Section 3, it is part of the global optimization over a longer time horizon as, for example, one walking step of the robot.

4.2.4 Cost Functions

The cost function design is explained in more detail in Schwienbacher et al (2011); Hildebrandt et al (2014). It is a weighted sum of costs which can be summarized as

$$\begin{aligned} L_y =& c_{jl}L_{jl} + c_{coll}L_{coll} + \\ & c_{cmf}L_{cmf} + c_{vel}L_{vel} + c_{ang}L_{ang} \end{aligned} \tag{28}$$

The costs $c_{jl}L_{jl}$ and $c_{coll}L_{coll}$ are devoted to penalizing violations of boundary conditions such as joint-limits resp. self-collision and collisions with the environment. $c_{cmf}L_{cmf}$ is a cost to penalize the deviation of the robot's motion from a desired comfort pose. While $c_{jl}L_{jl}$ contributes only to L_y when joint angles reach a joint-limit, $c_{cmf}L_{cmf}$ keeps the joint angles close to desired positions. $c_{vel}L_{vel}$ is devoted to reduce angular velocities. We choose the weights c_i taking physical dimensions of the costs into account to ensure that the boundary conditions are met. The kinematic structure of humanoid robots represents a particularity compared to conventional manipulators: its arms are open kinematic chains without defined workspace motions of their end effectors. Schwienbacher et al (2011) presented a method to use the arms DoF to compensate for angular momentum. $c_{ang}L_{ang}$ represents the corresponding cost.

4.3 Integration

Fig. 8 gives an overview of the integration of the *model-predictive kinematic planning* in the *global control.* The

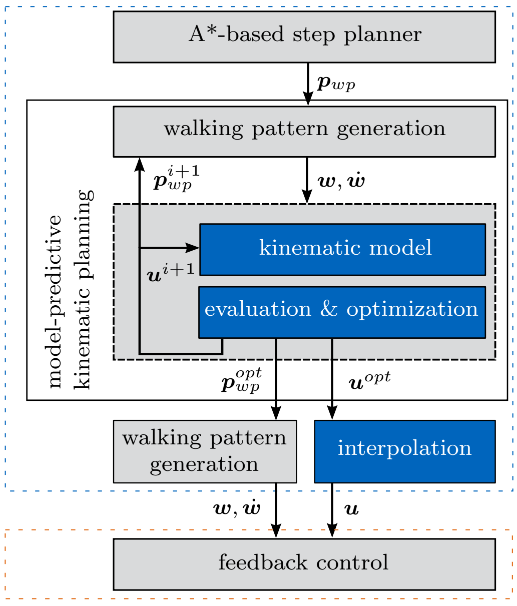

A*-based step planner
$\boldsymbol{p}_{wp}$
walking pattern generation
$\boldsymbol{p}_{wp}^{i+1}$
$\boldsymbol{w}, \dot{\boldsymbol{w}}$
model-predictive kinematic planning
kinematic model
$\boldsymbol{u}^{i+1}$
evaluation & optimization
$\boldsymbol{p}_{wp}^{opt}$
$\boldsymbol{u}^{opt}$
walking pattern generation
interpolation
$\boldsymbol{w}, \dot{\boldsymbol{w}}$
$\boldsymbol{u}$
feedback control

Fig. 8: Schematic of predictive planning for combined parameter and nullspace optimization.

output of the *model-predictive kinematic planning* is dependent on the method applied - either an optimized parameter set $\boldsymbol{p}_{wp}^{opt}$ or an optimized continous input vector $\boldsymbol{u}^{opt}$ or both (similar to the different methods tested for the simple manipulator in Section 3). The $\boldsymbol{p}_{wp}^{opt}$ is directly used to calculate the desired walking pattern, which is the input as $\boldsymbol{w}, \dot{\boldsymbol{w}}$ to the *feedback control.* $\boldsymbol{u}^{opt}$ is used as a feed-forward term to (23). Since the model used in the *model-predictive kinematic planning* and the real robot do not perfectly correspond, we add a drift compensation similar to the drift compensation used in the inverse kinematics for the end effector position to account for numeric drift. The resultant input vector is calculated as

$$\boldsymbol{u} = \boldsymbol{u}^{opt} + K_u(\boldsymbol{q}_M - \boldsymbol{q}_R) \tag{29}$$

with the vector $\boldsymbol{q}_M$ and $\boldsymbol{q}_R$ representing the joint position of the model resp. the real robot and the matrix K_u, which is positive-definite. We call this control architecture *model-predictive planning* in contrast to *model-predictive control.* The basic idea of our hierarchical control framework is that we plan the desired motion in advance. Underlying control layers modify the desired motion to account for disturbances. Since each control layer takes into account and improves the result of the previous one, the whole system becomes very robust.

4.3.1 Initial Solution - Kinematic Evaluation

The *Model-Predictive Kinematic Planning* depends on an initial kinematic movement which is executable. The initial parameter set $\boldsymbol{p}_{wp}$ is set heuristically by the *A*-based Step Planner* or by the user. It only approximates the full motion kinematics. For this reason, initially chosen $\boldsymbol{p}_{wp,init}$ may lead to kinematically infeasible movements. In Hildebrandt et al (2016) we introduced the *Kinematic Evaluation.* By integrating the kinematic model parametrized by $\boldsymbol{p}_{wp}$ and analyzing the resulting kinematics, the values of $\boldsymbol{p}_{wp}$ that would lead to kinematically non-feasible movements can be identified. Additionally, we proposed establishing an advanced error handling. Using another set of initial parameters including different footholds, the robot's movement is analyzed again and can be corrected. We get

$$\boldsymbol{p}_{wp,k+1} = \text{re-init}(\boldsymbol{p}_{wp}) \tag{30}$$

with the re-initialisation of $\boldsymbol{p}_{wp}$. The re-initialisation of $\boldsymbol{p}_{wp}$ is chosen based on $\boldsymbol{x}_{RCA}$. $\boldsymbol{x}_{RCA}$ denotes the projection of the local collision and kinematic limits avoidance on the workspace and is therefore an indicator of which parameter is limiting the movement. In this context, the challenge is to establish an interaction of the planning modules, step planning and trajectory planing which works reliably and fast enough to meet the real-time requirements.[4]

4.3.2 Gradients

A main difficulty in solving the combined optimal control problem is the computation of the derivatives of the Hamilton function. In general, the gradients could be computed by deriving an analytical expression of the gradients which is solved in each time step or, alternatively, applying a finite-difference scheme. As a third option, the *Efficient Gradient Computation* was presented by Toussaint et al (2007), which is particulary suitable to parameter-dependent derivatives and which

[4] The whole planning process has to be done in less than T_{Step}.

has been used in Hildebrandt et al (2016). In this approach, the gradients are re-formulated by the chain rule and then computed by forward integration from a known initial value. In our case, gradients with respect to the nullspace control input $\boldsymbol{u}(t)$ can be derived analytically, as shown in Schuetz et al (2014). Furthermore, a finite-difference scheme for a continuous trajectory would fail in real-time application due to its high computational expense. The formulas are presented in App. B.1. Further, derivatives with respect to step parameters $\boldsymbol{p}_{wp}$ can be computed by either the application of a finite-difference scheme or the *Efficient Gradient Computation*. The latter comes with very low computational expense since we have a limited set of parameters which are constant over the time of the motion.

4.3.3 Implementation Details

Fig. 9 shows the multi-process and multi-thread software architecture of Lola's real-time walking control system. Each stepping motion is analyzed and optimized before it is executed. In consequence, the planning time for the whole planning process of step k, including the calculation of the footholds and the planning of the desired walking pattern, must be accomplished during the step time of the previous step $T_{Step,k-1}$ in the real-time application (see Fig. 9). $T_{Step,k-1}$ varies between 0.7...1.2s, since it is user de-

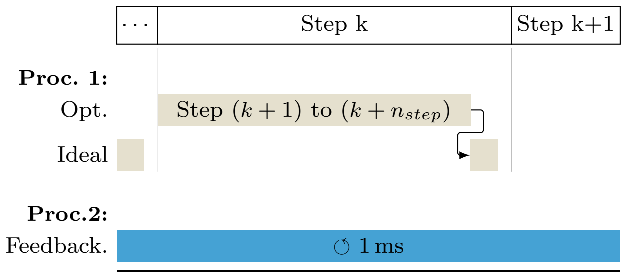


Fig. 9: Multi-process and multi-thread software architecture of Lola's real-time walking control system. Process 1 corresponds to the *Global Control*; process 2 corresponds to the *local control*.

pendent. Additionally, the step planner adapts T_{Step} according to the desired velocity of the robot and the environment. To handle the varying hard time constraints, we introduce an advanced time management. Based on $T_{Step,k-1}$ each step a maximal number of iterations can be estimated. It takes the time which is needed for one integration of the model into account depending on the current number of obstacles and on the necessity of the gradient calculation. The processing time for one integration with and without gradient calculation is summarized in Tab. 2. Since only a limited number of iter-

Table 2: Summary of maximal computation time for integration of kinematic model over one physically step. Comparison of gradient computation for the different methods and the integration of kinematic model without gradient computation. Runtimes are obtained from the real-time QNX computer.

method	max. [μs]
without gradient calculations	75
Numeric Gradient	75 n_{param}
Gradient of Nullspace	500

ations is possible, in most cases it is sufficient to calculate the descent of the cost function once and apply a line search or a interval nesting method. That way, the time-consuming integration including gradient calculation has to be done only once. The real-time application of our algorithm heavily depends on the possibility of aborting the optimization process at any given time. In the worst case, the optimization process is not able to converge to a solution superior to the initial one. In this case, only the kinematic feasibility is checked. Furthermore, we analyzed different integration step sizees. The analysis showed that an integration step size which is six times higher than the control cycle time still seems to be an adequate trade-off between accuracy of the kinematic movement of the robot and speed of the integration. Adaptable step lengths depending on the collision gradients showed bad results, which is in line with Betts (1998). A larger integration step size results in a coarse input vector $\boldsymbol{u}$. To be applicable to the *feedback control* we interpolate between the discrete set points $\boldsymbol{u}_k$ at time step t_k using third order polynomials. That way, the input $\boldsymbol{u}$ to the *feedback control* is no longer optimal, but the methods become applicable to the real system. This is represented as *interpolation* in Fig. 8.

5 Results

We analyzed the proposed methods in simulation and experiments with the robot *Lola*. A video showing the simulations and the experiments can be found at `https://youtu.be/twfDcsQvNBY`.

5.1 Simulation

The multi-body simulation used to verifiy the implemented methods includes unilateral and compliant contacts, motor dynamics as well as the joint control loops. For details see Buschmann (2010). We chose two different test cases to analyze the methods in this article - walking a step sequence and stepping over a complex obstacle.

5.1.1 Step Sequence

In the first test case, *Lola* executes a simple straightforward step sequence in an obstacle-free environment. The step lengths vary between $L_x = 0.3...0.4$m. We validate this test case in experiments as well (see Subsection 5.2.1). The resulting total costs of the combined optimization and of the local method are depicted in Fig. 10. We observe a strong cost reduction. Since the walking movement is a periodic and symmetric motion, results are very similar in each step. Fig. 11 shows the costs L_{jl} and $L_{self,coll}$ over the iterations of the gradient method. The optimization converges after two iterations for both L_{jl} and $L_{self,coll}$. Since possible collisions may lead to a failure of the whole system, collision costs are strongly weighted in the total costs. For this reason the reduction in the total costs stems mainly from the reduction of $L_{self,coll}$ resulting from the activation distance between arms and hip of the robot. Nevertheless, the other cost functions show moderate reductions as well. Only the comfort costs are slightly increased, since the arm motions have to be intensified to avoid the self-collision activation distance.

5.1.2 Stepping Over a Complex Obstacle

In the second test case we adapted a scenario from Hildebrandt et al (2016). It includes stepping over a complex shaped obstacle as depicted in Fig. 13 in the collision world representation. This test case forces the robot to perform a kinematically complex stepping-over motion. It has been shown in Hildebrandt et al (2016) that a model-predictive *kinematic parameter evaluation & optimization* significantly improves the chance of success for the biped motion planning. Similar to the first test case, the combined optimization reduces the costs of $L_{self,coll}$ through nullspace optimization. That way, the chance of collisions between arms and hip of the robot is reduced. Using only the *local control* approach collisions are likely to occur as seen in Fig. 13a. Fig. 14 shows the cost reduction during the three relevant steps of the overstepping motion over the iterations. Almost full cost reduction can be achieved with the first iteration of the gradient method, which includes backtracking line-search. Tab. 3 shows the resulting parameters. The modifications show that the heuristic of the *A*-based step planner* provides good initial values, but not optimal ones. In such complex scenarios, the predictive approach is superior to the heuristic approach.

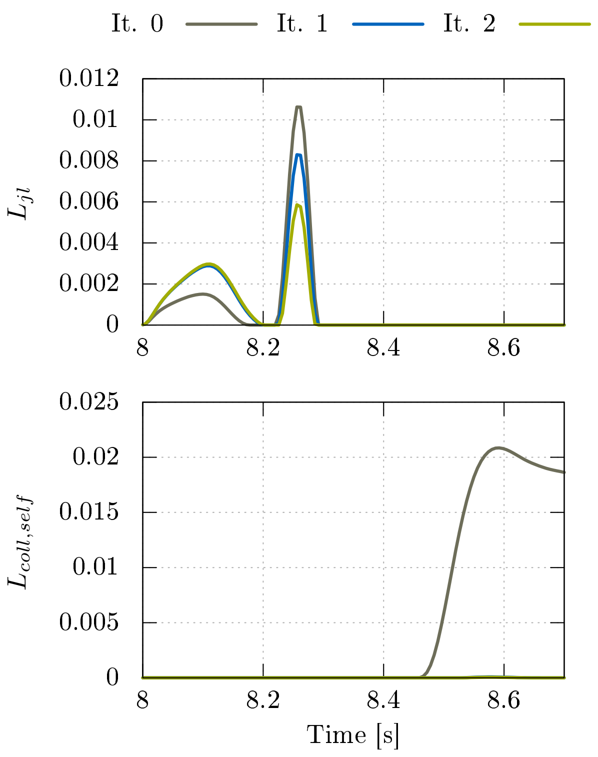


Fig. 11: Optimization of joint limit & self-collision costs

Table 3: Opimized parameters for stepping over a complex obstacle.

	Initial		Optimized	
Step	H_{cog}	dz_{step}	H_{cog}	dz_{step}
4	$0.880m$	$0.112m$	$0.885m$	$0.112m$
5	$0.880m$	$0.112m$	$0.882m$	$0.120m$
6	$0.880m$	$0.112m$	$0.885m$	$0.120m$

5.2 Experiment

To validate the promising simulation results we validated the methods in experiments. Again, we present different scenarios to measure the performance and the real-time ability of our software framework. First, the results of a step sequence similar to the sequence in

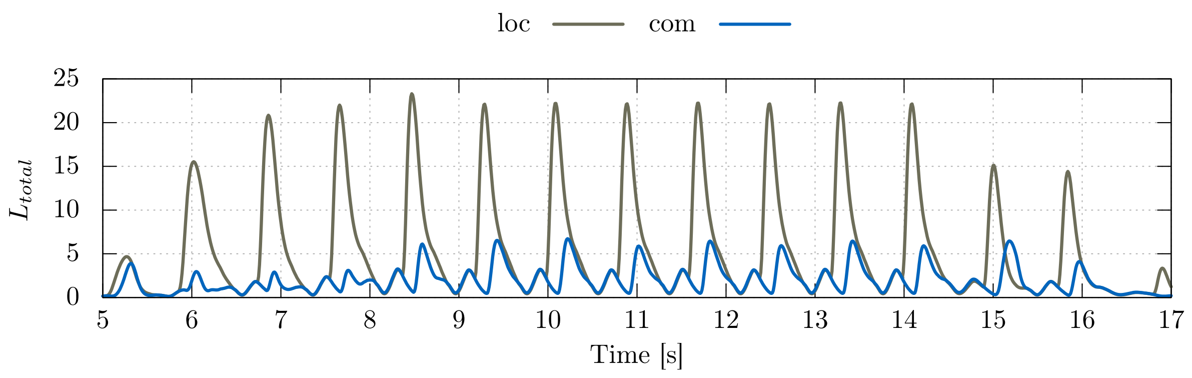


Fig. 10: Simulation results for step sequence. Total costs for local and combined optimization.

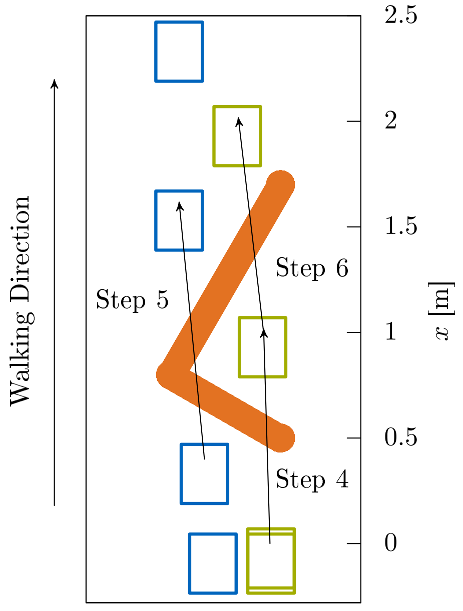


Fig. 12: Step sequence for simulation test case with complex obstacle.

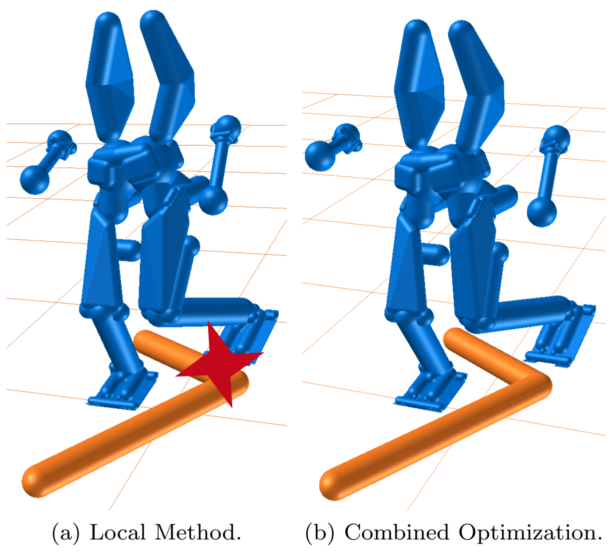


Fig. 13: Collision world representation of complex scenario. *Lola*'s collision model (blue), while stepping-over obstacle (orange) for local method and for combined optimization.

Subsection 5.1.1 are presented. Then, the real-time capability is validated in experiments in a cluttered environment with a vision system.

5.2.1 Step Sequence

The first experiment includes a simple step sequence in an environment without any obstacles (see Subsection 5.1.1). *Lola* executing the sequence is depicted in Fig. 15. The robot walks a distance $l_{exp} = 5$m with a step length $L_x = 0.3 \ldots 0.4m$ and a step height of $dz_{clear} = 0.04$m. The real-time requirement forces us to limit the optimized parameters to $n_{opt} = 2$. Due to recent work presented in Hildebrandt et al (2016), we chose the parameters dz_{step} and dy, since they seem to be relevant in kinematically challenging situations. We directly measure joint positions and velocities to compute the cost functions. The measured costs in Fig. 17 show a comparison of the local and the combined optimization. The results resemble the simulated case (for example the strong reduction of $L_{coll,self}$), yet it is not of the same quantity. The reason for this is the smaller number of iterations in the gradient optimization given by the real-time requirements of the walking control and the difference due to sensor feedback in the *Feedback Control*. Fig. 16 shows a frontal view on *Lola* walking with and without combined optimization. It clearly shows the effect of the reduced $L_{coll,self}$ by the arm motions. As expected, in this step sequence experiment the lateral swing parameter dy is not changed

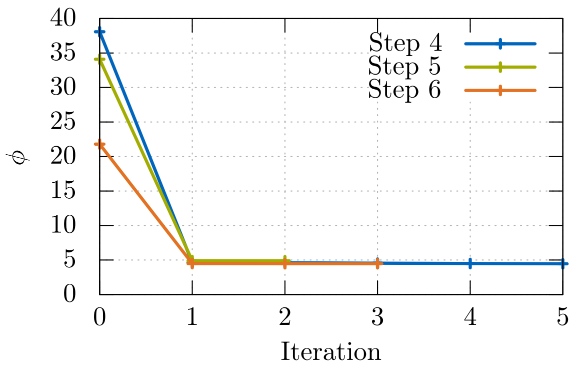


Fig. 14: Cost over iterations for combined optimization. Gradient method includes backtracking line-search.

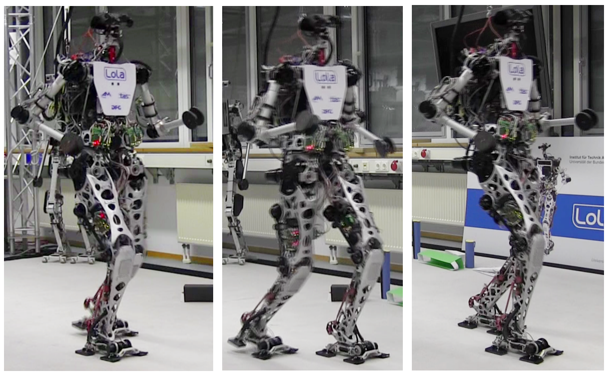


Fig. 15: Experimental results: *Lola* executing step sequence with combined optimization.

by the optimization while the step height is decreased. This reduces the joint velocities. Fig. 18 shows a comparison of L_{total} for the periodic walking motion of all presented methods - local method, parameter optimization, nullspace optimization and combined optimization. It is clearly visible, that the combined optimization method outperforms the others. The comparison between nullspace optimization and combined optimization illustrates that for kinematically simple motion the cost reduction can be mainly attributed to an optimized nullspace exploitation.

5.2.2 Cluttered and Unknown Environment

In addition to the experimental setups with obstacle-free environment we present the results for an experiment in an unknown and cluttered envrionment. The experimental setup and its approximation for the robot control is depicted in Fig. 19. Due to the complexity of the perception and planning modules of *LOLA*, it is hard to conduct experiments in cluttered environments which can be reproduced identically for all methods. Little changes in the environment setup lead to major

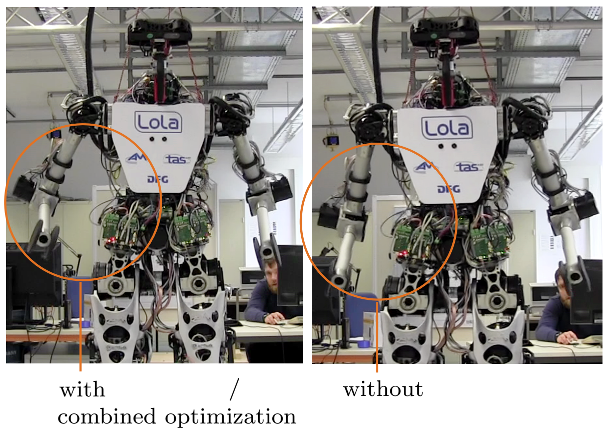


Fig. 16: Nullspace exploitation: *Lola* walking with and without combined optimization. Increased distance between arms and hip for $L_{coll,self}$ reduction.

changes in step planning and subsequently in motion planning. The main objective of this experiment is not to compare the results of all methods but to show their real-time capacities. Although we tested all methods in cluttered environment, we show here only the results of the combined optimization method. The experiment is highly demanding in terms of the real-time capability. Since *Lola*'s field of view is limited, the *vision system* perceives with a frame rate of 30 ms new obstacles which are taken into account in the motion planning algorithms. The motion planning adapts the step time of each step and the kinematic motion. Furthermore, the integration time of the model-predicitve methods is obstacle-dependent. Therefore, it has to be possible to adapt planning time according to the current walking scenarios and to abort the optimization methods at any given time. Fig. 20 shows L_{total} for each step for the combined optimization and the local method. For most steps, the combined optimization method could reduce the total costs. In steps 14, 15, 18, 22 and 23 the optimization had to be aborted because of the limited planning time. A picture of *Lola* navigating among the previously unknown obstacles is shown in Fig. 21.

6 Conclusion and Outlook

In this paper, we present methods for real-time motion generation of redundant robots. We validate their performance in various experiments in unknown environments with our bipedal robot *Lola*. The methods based on a predicitve model approach are introduced using a minimal model of a redundant robot. Our methods presented in Hildebrandt et al (2016); Schuetz et al (2014) for different applications are applied and compared to

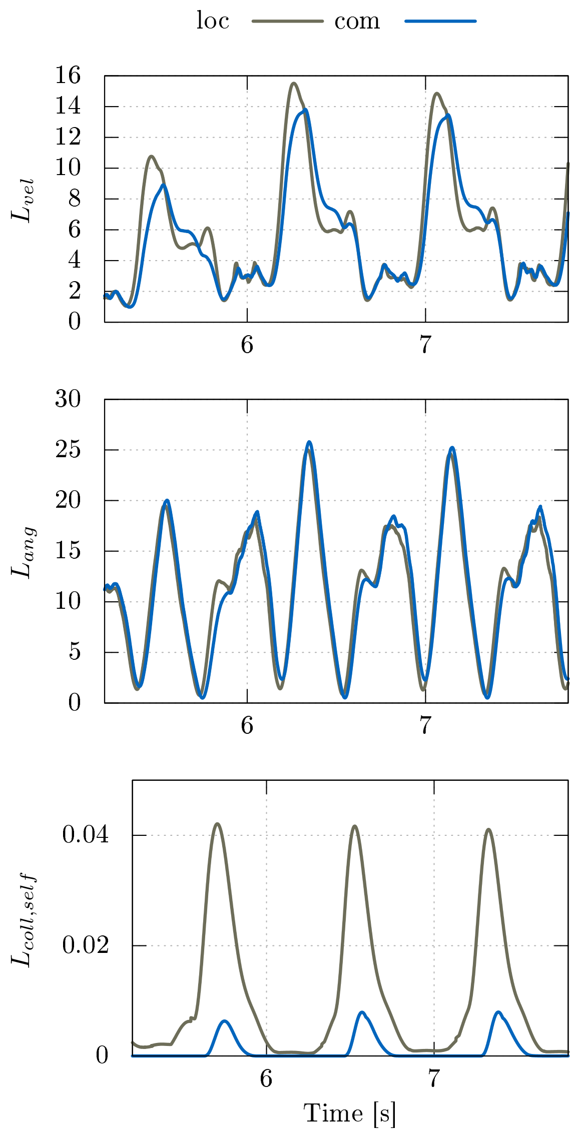


Fig. 17: Data of step sequence experiment: costs $(L_{vel}, L_{ang}, L_{coll,self})$ over time.

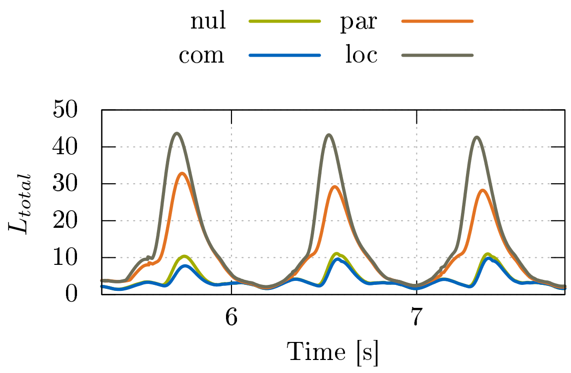


Fig. 18: Data of step sequence experiment: total costs for local method, parameter optimization, nullspace optimization and combined optimization.

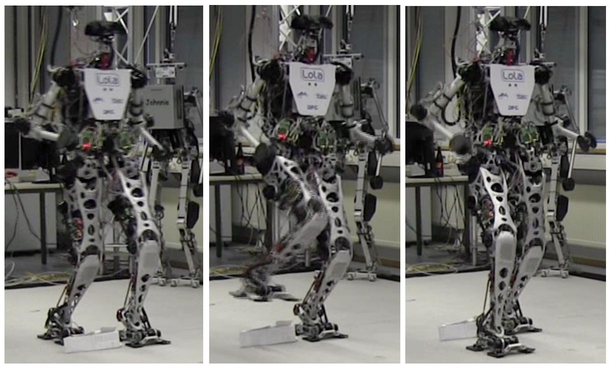


Fig. 19: Experimental results: *Lola* executing step sequence with combined optimization in environment with obstacles. Figure shows *Lola* stepping over an obstacle.

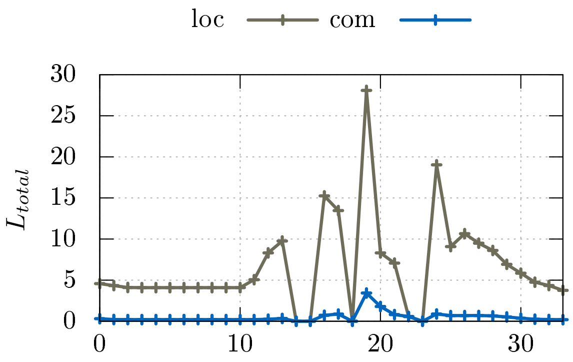


Fig. 20: Experimental results for cluttered environment: comparison between total costs resulting from combined optimization and from *local control*.

a reference motion calculated by the well-known *Automatic Supervisory Control*. Furthermore, we present a novel method to combine optimization of parameterized workspace trajectories and the continuous input to the redundancy resolution. The methods are integrated in our control framework for bipedal locomotion and adapted to the characteristics of humanoid walking. We analyze them in simulations and successfully conducted experiments. The methods significantly improve *Lola*'s performance and allow for complex motions in real-time.

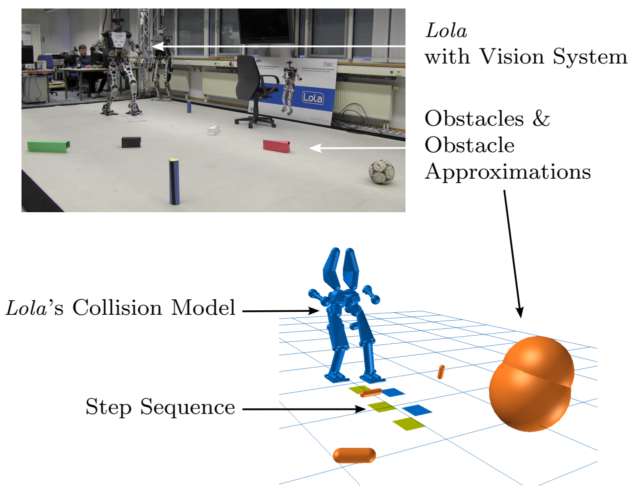


Fig. 21: Experimental setup for experiments in cluttered environment.

Limitations and Outlook

The presented methods separately or combined allow for kinematic motions in complex walking scenarios, while having dynamically feasible movements. The calculated motions are optimal in respect to our defined cost functions. But bipedal locomotion is also more than kinematically optimal motions. Although complex motions are feasible when applying the presented method, we can not answer the question "What is optimal bipedal locomotion?". In our simulations and experiments the cost function design plays a crucial role for the performance of the optimization methods. In the current implementation, the influence of the collision avoidance is rather high. Zucker et al (2013) propose an approach of *cost weight scheduling* to dynamically adapt the cost weighting. The influence of the collision avoidance on the total costs is increased near obstacles. This results in better behavior of the gradient method and could also improve the results in our application. Another possibility would be to introduce a strict task hierarchy as described in Ott et al (2015). In experiments, the real-time restriction severely limits the performance of the optimization methods for redundancy resolution and the combined method. In the current implementation, only few iterations are possible due to the time-consuming gradient calculations. The performance of the optimization methods in real applications can be improved by using more threads for cost evaluations due to the method's real-time character. We intend to exploit this software-related possibility in future work. In the current implementation, the CoM trajectory is parameterized with one parameter per step. A more sophisticated trajectory representation may help to improve the interaction between the workspace trajectories determining the kinematic motions of *Lola*. In the future, we are planning to apply our methods also on our redundant manipulator (see Schuetz et al (2014)) and analyze its performance for agricultural applications in cluttered environments.

A Acknowledgements

We would like to acknowledge the DAAD and the *Deutsche Forschungsgemeinschaft* (DFG - project BU 2736/1-1) for their support of this project.

B Appendix

B.1 Gradients for Optimization of Redundancy

Considering the function $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}, p) = \dot{\boldsymbol{q}}$ as shown in (10) and the *Moore-Penrose* pseudoinverse $\boldsymbol{J}^{\#} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}$, the analytical gradients necessary for redundancy optimization can be formulated as follows:

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}} = \frac{\partial \dot{\boldsymbol{q}}}{\partial \boldsymbol{q}} = \frac{\partial \boldsymbol{J}^{\#}}{\partial \boldsymbol{q}} \dot{\boldsymbol{w}} - \left(\frac{\partial \boldsymbol{J}^{\#}}{\partial \boldsymbol{q}} \boldsymbol{J} - \boldsymbol{J}^{\#} \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}}\right) \boldsymbol{u} \tag{31}$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} = \boldsymbol{I} - \boldsymbol{J}^{\#} \boldsymbol{J} \tag{32}$$

$$\begin{aligned} \frac{\partial \boldsymbol{J}^{\#}}{\partial \boldsymbol{q}} = \frac{\partial \boldsymbol{J}^T}{\partial \boldsymbol{q}} \left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1} + \boldsymbol{J}^T \Big[-\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1} \ldots \\ \left(\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \boldsymbol{J}^T + \boldsymbol{J} \frac{\partial \boldsymbol{J}^T}{\partial \boldsymbol{q}}\right) \left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1} \Big] \end{aligned} \tag{33}$$

$$\left(\frac{\partial L_{cmf}}{\partial \boldsymbol{q}}\right)^T = 2\left(\boldsymbol{q} - \boldsymbol{q}_{cmf}\right) \tag{34}$$

$$\left(\frac{\partial L_{vel}}{\partial \boldsymbol{q}}\right)^T = 2\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^T \dot{\boldsymbol{q}} \tag{35}$$

$$\left(\frac{\partial L_{vel}}{\partial \boldsymbol{u}}\right)^T = 2\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)^T \dot{\boldsymbol{q}} \tag{36}$$

with $\boldsymbol{q}_{cmf}$ a user defined comfort pose. For cost gradients regarding collision avoidance and joint limits, we refer to Buschmann et al (2009) and Schuetz et al (2014) respectively.

References

A Liegeois (1977) Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. IEEE Transactions on Systems (12):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

Baudouin L, Perrin N, Moulard T, Lamiraux F, Stasse O, Yoshida E (2011) Real-time replanning using 3D environment for humanoid robot. In: IEEE-RAS International Conference on Humanoid Robots

Betts JT (1998) Survey of Numerical Methods for Trajectory Optimization. Journal of Guidance, Control, and Dynamics 21(2):193–207

Bocek M (1980) Conjugate gradient algorithm for optimal control problems with parameters. Kybernetika 16(5):454–461

Buschmann T (2010) Simulation and Control of Biped Walking Robots. PhD thesis, Technical University of Munich

Buschmann T, Lohmeier S, Ulbrich H, Pfeiffer F (2005) Optimization based gait pattern generation for a biped robot. In: IEEE-RAS International Conference on Humanoid Robots

Buschmann T, Lohmeier S, Bachmayer M, Ulbrich H, Pfeiffer F (2007) A Collocation Method for Real-Time Walking Pattern Generation. In: IEEE/RAS International Conference on Humanoid Robots

Buschmann T, Lohmeier S, Ulbrich H (2009) Biped walking control based on hybrid position/force control. In: IEEE-RSJ International Conference on Intelligent Robots and Systems

Buschmann T, Lohmeier S, Schwienbacher M, Favot V, Ulbrich H, von Hundelshausen F, Rohe G, Wuensche HJ (2010) Walking in Unknown Environments - A Step Towards More Autonomy. In: IEEE-RAS International Conference on Humanoid Robots

Buschmann T, Favot V, Lohmeier S, Schwienbacher M, Ulbrich H (2011) Experiments in Fast Biped Walking. In: IEEE International Conference on Mechatronics

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

Chitta S, Sucan I, Cousins S (2012) MoveIt! [ROS Topics]. IEEE Robotics & Automation Magazine 19(1):18–19

Englsberger J, Ott C, Roa MA, Hirzinger G (2011) Bipedal Walking Control Based on Capture Point Dynamics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems

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

Gienger M, Toussaint M, Jetchev N, Bendig A, Goerick C (2008) Optimization of Fluent Approach and Grasp Motions. In: IEEE-RAS International Conference on Humanoid Robots

Guan Y, Yokoi K, Tanie K (2005) Feasibility: Can Humanoid Robots Overcome Given Obstacles? In: IEEE International Conference on Robotics and Automation

Guan Y, Yokoi K, Tanie K (2006) Stepping over obstacles with humanoid robots. IEEE Transactions on Robotics 22(5):958–973

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

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

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

Hildebrandt AC, Wittmann R, Sygulla F, Wahrmann D, Rixen D, Buschmann T (2017) Versatile and Robust Bipedal Walking in Unknown Environments (submitted). IEEE Transactions on Robotics

Hornung A, Bennewitz M (2012) Adaptive Level-of-Detail Planning for Efficient Humanoid Navigation. In: IEEE International Conference on Robotics and Automation

Kajita S, Kanehiro F, Kaneko K, Fujiwara K, Harada K, Yokoi K, Hirukawa H (2003) Biped walking pattern generation by using preview control of zero-moment point. In: IEEE International Conference on Robotics and Automation, IEEE, pp 1620–1626

Koch KH, Mombaur K, Stasse O, Soueres P, Koch KH, Mombaur K, Stasse O, Optimization PS (2014) Optimization based exploitation of the ankle elasticity of HRP-2 for overstepping large obstacles. In: IEEE-RAS International Conference on Humanoid Robots

Lohmeier S, Buschmann T, Ulbrich H (2009) System Design and Control of Anthropomorphic Walking Robot LOLA. IEEE/ASME Transactions on Mechatronics 14(6):658–666

Nakamura Y, Hanafusa H (1987) Optimal Redundancy Control of Robot Manipulators. The International Journal of Robotics Research 6(1):32–42

Nishiwaki K (2011) Online design of torso height trajectories for walking patterns that takes future kinematic limits into consideration. In: International Conference on Control, Automation and Systems

Nishiwaki K, Chestnutt J, Kagami S (2012) Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor. The International Journal of Robotics Research 31(11):1251–1262

Ott C, Dietrich A, Albu-Schäffer A (2015) Prioritized multitask compliance control of redundant manipulators. Automatica 53:416–423

Schuetz C (2017) Trajectory Planning for Redundant Manipulators. PhD thesis, Technical University of Munich

Schuetz C, Buschmann T, Baur J, Pfaff J, Ulbrich H (2014) Predictive Online Inverse Kinematics for Redundant Manipulators. In: IEEE International Conference on Robotics and Automation

Schulman J, Ho J, Lee A, Awwal I, Bradlow H, Abbeel P (2013) Finding locally optimal, collision-free trajectories with sequential convex optimization. In: Robotics: Science and Systems

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

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

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, Stryk OV (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

Takenaka T, Matsumoto T, Yoshiike T (2009) Real time motion generation and control for biped robot - 3rd report: Dynamics error compensation-. In: IEEE/RSJ International Conference on Intelligent Robots and Systems

Tassa Y, Erez T, Todorov E (2012) Synthesis and stabilization of complex behaviors through online trajectory optimization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems

Toussaint M, Gienger M, Goerick C (2007) Optimization of sequential attractor-based movement for compact behaviour generation. In: IEEE-RAS International Conference on Humanoid Robots

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

Verrelst B, Stasse O, Yokoi K, Vanderborght B (2006) Dynamically Stepping Over Obstacles by the Humanoid Robot HRP-2. In: IEEE-RAS International Conference on Humanoid Robots

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

Whitney D (1969) Resolved motion rate control of manipulators and human prostheses. IEEE Transactions on Man-Machine Systems 10(2):47–53

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

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

Zucker M, Ratliff N, Dragan aD, Pivtoraiko M, Klingensmith M, Dellin CM, Bagnell Ja, Srinivasa SS (2013) CHOMP: Covariant Hamiltonian optimization for motion planning. The International Journal of Robotics Research 32(9-10):1164–1193