Nonsmooth Trajectory Optimization for Wheeled Balancing Robots With Contact Switches and Impacts

Victor Klemm, *Graduate Student Member, IEEE*, Yvain de Viragh, *Graduate Student Member, IEEE*, David Rohr, Roland Siegwart, *Fellow, IEEE*, and Marco Tognon, *Member, IEEE*

***Abstract*—Recent years have seen a steady rise in the abilities of wheeled–legged balancing robots. Yet, their use is still severely restricted by the lack of efficient control algorithms for overcoming obstacles such as stairs. We take a considerable step toward closing this gap by presenting a fast trajectory optimizer for generating trajectories over a large class of challenging terrains. By limiting the underlying modeling to the planar, nonlinear rigid-body dynamics and subdividing the terrain into contact-phases, a tractable nonlinear programming problem is obtained. The model explicitly accounts for contact switches and impacts, traction limits, and actuation bounds. By introducing an arc-length-related parametrization, the trajectories are rendered inherently contact constraint-consistent. We apply our method to the specific case of the wheeled bipedal robot *Ascento*, for which we derive closed-form expressions of the dynamics equations, including the kinematic loops. To track the trajectories, we propose a simple LQR-based controller. The approach is validated in real-world experiments where we show the execution of trajectories for traversing steps, driving up ramps, jumping, standing up, and driving up entire stairways. To the best of our knowledge, enabling the latter by means of trajectory optimization (TO) is a novelty for wheeled–legged robots.**

***Index Terms*—Contact modeling, legged robots, optimization and optimal control, wheeled robots.**

I. Introduction

VERSATILE, fast, and energy-efficient robots with rough-terrain capabilities would benefit a wide variety of tasks, ranging from search and rescue, to inspection, patrolling, and delivery. While purely legged systems have proven to be a reliable and robust solution [1], they typically have considerable speed limitations and significant cost of transport, especially when carrying payloads. Purely wheeled systems, on the other hand, do not have these shortcomings [2] but suffer from comparatively limited rough-terrain capabilities. Although solutions exist, e.g., transformable wheels [3], rover-like linkages [4], and tracks [5], they tend to severely compromise the systems' agility and speed.

Articulated wheeled–legged systems combine the best of both worlds. A wheeled version of the quadruped *ANYmal*, for instance, has demonstrated impressive locomotion capabilities [6], [7]. For an extensive survey of other quadrupedal wheeled–legged robots, see [8]. The only major downside of such quadrupedal realizations are the high number of potentially expensive actuators and the large footprint. These considerations

Fig. 1. *Top and bottom left:* The new *Ascento* robot climbing a stairway by repeatedly tracking motion trajectories optimized for a single step. The step height of the two stairs are 14.5 and 17 cm, respectively. *Bottom right:* Driving down a stair using only the proposed proprioceptive tracking controller. We remark that all motions shown in the *Ascento Pro* video,[1] which includes climbing the stair shown on the top, were performed using the TO scheme and/or the tracking controller proposed in this work.

Manuscript received 7 June 2023; accepted 12 September 2023. Date of publication 20 October 2023; date of current version 13 December 2024. This paper was recommended for publication by Associate Editor A. Dietrich and Editor E. Yoshida upon evaluation of the reviewers' comments. This work was supported by the Swiss National Science Foundation (SNF) through the National Centre of Competence in Research (NCCR) under Grant 188596. *(Corresponding author: Victor Klemm.)*

Victor Klemm was with the ASL, ETH Zürich, 8092 Zürich, Switzerland. He is now with the RSL, ETH Zürich, 8092 Zürich, Switzerland (e-mail: vklemm@ethz.ch).

Yvain de Viragh was with the ASL, ETH Zürich, 8092 Zürich, Switzerland (e-mail: yvaindeviragh@gmail.com).

David Rohr and Roland Siegwart are with the ASL, ETH Zürich, 8092 Zürich, Switzerland (e-mail: drohr@ethz.ch; rsiegwart@ethz.ch).

Marco Tognon was with the ASL, ETH Zürich, 8092 Zürich, Switzerland. He is now with the INRIA Rennes in the Rainbow team, 35042 Rennes, France (e-mail: mtognon@ethz.ch).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2023.3326334, provided by the authors.

Digital Object Identifier 10.1109/TRO.2023.3326334

[1] [Online]. Available: https://youtu.be/Uxt2wTI0m5o

make balancing wheeled–legged robots appear as attractive alternatives. Indeed, apart from research [9], [10], [11], [12], [13], there has also been notable interest into such systems from industry [14], [15], [16]. However, in order to actually leverage their potential and enable the traversal of typical human environments, appropriate control methods are required. The existing literature on wheeled–legged balancing robots is unsatisfactory in this regard since the respective control methods are very limited in their abilities to overcome obstacles.

Our work makes an important step toward filling this gap by presenting a trajectory optimization (TO) scheme explicitly accounting for the terrain and its validation in several experiments with the *Ascento* robot (see Fig. 1). The proposed TO scheme also marks a considerable improvement over our previous approach [17], which deals with uneven and unknown terrain by active compliance through hierarchical whole-body control with a built-in linear-quadratic regulator (LQR). It proved to be robust but, lacking any trajectory planning, limited to moderate terrain changes. In our introductory work to the *Ascento* robot [11], we furthermore showed that it is possible to overcome steps by executing a hand-crafted jumping routine. The resulting trajectories were, however, strongly suboptimal. The admissible step height was low, reliability was poor, and considerable room before and after the step was required for the robot to accelerate and decelerate, thus prohibiting the traversal of stairs.

The TO scheme presented in this work allows the robot to overcome these limitations and traverse a multitude of different terrains and, notably, entire standard stairways (with a step height of 17 cm and a step width of 29 cm). Enabling a wheeled–legged robot to do so by means of TO is, to the best of our knowledge, a novelty. We present the experiments on two improved *Ascento* prototypes. They feature stronger motors and larger wheels than the robots previously used, enabling them to negotiate typical urban environments. Thus, our work also showcases that wheeled–legged bipedal balancing robots are indeed excellent candidates to assist us in our daily environments.

A. Problem Description and Related Work

Locomotion of wheeled–legged robots requires the control algorithms to carefully leverage the system dynamics in order to keep balance while traversing challenging terrain—motor torques are limited, joints have only a certain range of motion, traction limits must be respected, and ground contact should not accidentally be made or lost. Furthermore, it is sometimes desired that the system trajectories should be optimal w.r.t. a criterion such as time or energy consumption. Accordingly, these challenging locomotion tasks cannot be solved in a purely reactive manner, but require to anticipate the terrain and plan future actions. Explicitly doing so amounts to model-based TO, which is (arguably) the most general way to address the problem. Based on a model of the dynamics, including the environment, the system trajectory is optimized such as to adhere to constraints and/or minimize a custom objective function. Thanks to the intuitive physical description, variations in the dynamics parameters, the terrain, or the objective function typically translate to straightforward modifications of the problem formulation. Moreover, a wealth of well-proven mathematical tools can be applied for theoretical analysis.

However, while theoretically sound and intuitive, solving the optimization problem in practice can quickly become intractable, requiring a careful problem setup, as we elaborate in the following. Alternatively to the explicit, general optimization of solution trajectories, the problem can also be tackled using data-driven methods, such as reinforcement learning (RL). These have recently emerged as strong competitors to classical control techniques and found successful application in the (wheeled–)legged robotics community. With sufficient training data and domain randomization, highly performant and robust solutions can be obtained, some of which seem beyond reach of classical TO. However, the difficulty to physically interpret their inner workings, the usually large amount of training (data), and the potential need to retrain if system parameters, the environment, or the objective change, are some of the downsides to the possibly high performance. In the following paragraphs we first provide an overview of the current state of the art in model-based TO and identify the research gap that we attempt to fill. To position our work within the context of the rapidly expanding capabilities of data-driven methods, we then elaborate on their comparative merits and the respective state of the art.

One may treat the general problem of generating trajectories for legged and wheeled–legged robots as an infinite-dimensional optimal control problem subject to various constraints. Among these constraints are the following.

1) The nonlinear dynamics equations, which render the problem nonconvex.
2) Bounds on inputs and states. These include pure state constraints, which impede treatment of the problem as one in the input variables, for instance through differential dynamic programming.
3) Complementarity constraints which, assuming hard contacts, enforce that either there be no contact forces or that the velocity normal to the contact surface be zero.

The latter suggest an inherently combinatorial nature of the problem and a computational complexity for finding a globally optimal solution that grows exponentially with the time horizon. Moreover, it is well-known that the complementarity constraints do not satisfy the linear independence constraint qualification (LICQ), since they induce a rank loss in the system of equations associated to the Karush–Kuhn–Tucker conditions (see [18], for instance). This complicates the matter not only from an analytical/theoretical point of view but also requires special care on the numerical solver side. Furthermore, nonsmooth terrain introduces an additional, mathematically closely related complication. It renders the majority of traditional approaches for simulating and controlling wheeled (inverted-pendulum) systems—e.g., [19], [20]—unsuitable for our purposes, because smooth terrain is assumed in the underlying modelings. Readers wishing to learn more about the topic of *nonsmooth mechanics* and its intricacies are pointed to classics such as [21], [22].

In short, even when seeking a locally optimal solution only, the problem is generally anything but trivial. This is also supported by empirical evidence. All optimization-based approaches for bipedal or quadrupedal robots known to us which attempt to model the problem in its generality, i.e., using a full rigid-body dynamics model and complementarity constraints, either do not state the computation times of the solver or report convergence times in the order of minutes to hours for trajectories with horizons in the order of seconds [23], [24], [25], [26], [27], [28], [29]. Moreover, except for the last one, all of these works limit themselves to planar robot models.

Not surprisingly, the majority of publications—including the present one—addresses these difficulties by trading-off generality, range of admissible motions, local optimality, or physical accuracy for convergence speed. For purely legged robots such approaches range from solving the problem for a system model linearized about a reference trajectory [30], resorting to soft-contact models [31], employing simplified dynamics models where the legs are treated as massless and contact-timings are predetermined [32], [33], [34], [35], [36], to applying heuristics-based[2] schemes centered around the *zero-moment point* concept [38], [39]. In most of these works, the terrain is moreover assumed to be flat.

The works [40], [41], which are concerned with purely legged robots, represent an exception in this regard and they deserve particular attention due to a close conceptional relation to our approach. Namely, they build upon a phase-based parametrization where the number of contact switches is predetermined, but where the contact timings—or, in other words, the length of the contact phases—are optimized. Contrasting this approach to the use of complementarity constraints, which result in a problem that may equivalently be written as an integer program, this has the considerable advantage of leading to a conventional nonlinear program (NLP), which satisfies the LICQ. Readers interested in a theoretically more elaborate treatment of phase-based problem formulations are referred to works treating switching-time optimization of hybrid systems, e.g., [42].

The reasoning in the preceding few paragraphs suggests the TO problem for wheeled–legged systems to be at least as difficult as the one for purely legged systems, since additional degrees of freedom (DOFs) at the contact points must be controlled and corresponding nonholonomic constraints must be respected. Indeed, those works demonstrating complex motions, such as hybrid driving and stepping, employ severe simplifications—similar to the ones mentioned for purely legged systems—to achieve reasonable computation times [6], [7], [43], [44], [45], [46], [47], [48], [49]. Of these, only [7], [48], [49] explicitly take the terrain into account, the latter two limiting themselves to pure driving. The recently published work [7] builds (similarly to [48]) upon the phase-based approach [40] and extends it to wheeled systems by relaxing the zero-velocity constraint on feet in contact with the ground such that motion along the torso's heading direction and perpendicular to the terrain normal is permitted. This is in strong contrast with, among others, the arc-based minimal state representation adopted in our work, which is inherently contact constraint-consistent. A further important difference is the approximation of the robot by a 3-D floating base with point feet/wheels, whereas we employ a planar version of the full rigid-body dynamics. Our choice is motivated by the observation that, under the mild assumption of zero lateral tilting, the chosen plane locally matches the usually primary operating direction of wheeled–legged balancing robots (walking laterally is typically not possible). That is, the plane is aligned with the direction of driving and balancing. We thus argue that trading-off generality of the 3-D modeling for increased accuracy of the planar one is a sensible choice.

Yet a different approach to overcoming challenging terrain is taken for rover-like systems such as [50], [51], [52]. In these cases the optimization solves for contact forces in a static fashion without any TO, which renders this approach unsuitable for balancing systems.

Turning our attention to the specific case of wheeled–legged balancing robots, we observe that recent approaches typically rely on modified, instantaneous feedback laws, such as an LQR, which are often applied in combination with a hierarchical optimization for task-space control of the upper body [11], [13], [17], [53], [54], [55]. Some TO approaches exist, but they either omit a model with contact switches in the optimization [56], or are developed for specific motions, such as jumping [12], [57]. Unfortunately, little is known about the control approach of *Boston Dynamics*' *handle* robots [14], [15], which demonstrated impressive performance.

To summarize, the topic of model-based TO for wheeled–legged robots, with the terrain explicitly accounted for, has been explored only marginally in the literature. The mathematical nature of the problem makes it a challenging one and existing approaches either suffer from poor convergence times, very restrictive assumptions, or are tailored to quadrupedal robots rather than balancing ones. Our work attempts to remedy this research gap for wheeled–legged balancing robots.

We now turn our attention to data-driven approaches, such as RL. Despite the lack of theoretical guarantees, the resulting controllers have recently demonstrated unprecedented robustness. Namely, they have proven to be very effective at preventing robots from falling, both for purely legged [58], [59], [60] and wheeled–legged ones [61], [62], [63], [64], [65].

Such controllers have also demonstrated the ability to handle and exploit complex contact scenarios, where not only the feet are in contact with the ground but also parts such as the knees and the torso [66]. Some of these latter motions currently do not seem within the reach of computationally tractable model-based TO schemes. On the other hand, when it comes to traversing unknown terrains where planning ahead is required and the system dynamics must be properly leveraged, both model-based TO [32], [33] and RL [67] approaches have recently demonstrated increasingly satisfactory performances, at least for the case of quadrupedal systems. Although performance and versatility of RL-based methods are steadily increasing, the generation of longer, more involved motions still remains a challenging problem. The main difficulty may typically be considered finding a structured way of exploring the solution space.

[2]For instance, Raibert's criterion [37] is a frequently used heuristic to predict foothold locations.

This holds in particular for common RL methods, which rely on sampling-based, zeroth-order gradient estimates. By contrast, TO-methods can target such problems with remarkable sample efficiency, making them valuable also as data generators that can considerably accelerate RL training by providing guidance for the exploration of complex motions.

B. Contributions and Outline

Our main contribution is a general TO scheme for locomotion of wheeled–legged balancing robots over uneven terrain. Its performance and novelty stems in particular, but not only, from the combination of the following characteristics.

1) Limiting the TO to the planar, nonlinear rigid-body dynamics to reduce problem dimensionality and obtain a tractable model (see Section II).
2) Subdivision of the terrain into contact phases, whereby the discrete nature and exponential complexity of typical mode scheduling problems is avoided (see Section II-A).
3) Arc-length related parametrization of the wheel state which permits to formulate the dynamics of contact phases in a kinematically inherently constraint-consistent subspace (see Section II-B).
4) Explicit formulation of contact force constraints and impulsive impacts through a reprojection of the dynamics into the full—kinematically not inherently constraint-consistent—space (see Sections II-C and II-D).
5) Transcription into an NLP which can be solved by conventional solvers in a matter of seconds. The chosen formulation allows for branchless (i.e., without conditional statements) use of autodifferentiation (see Section III).
6) A custom augmentation of the Hermite–Simpson collocation scheme which provides improved accuracy without requiring additional function evaluations. It is applicable to systems where the generalized velocities are the time derivative of the generalized coordinates (see Section III-C).

Furthermore, as part of the application to the *Ascento* robot, our contribution includes the following.

1) We show how to derive analytic expressions of *Ascento*'s kinematic loops which are closed-form and valid over the entire operating range. This allows us to obtain smoothly differentiable dynamics equations (see Section IV-A).
2) We propose a simple, intuitive high-gain proportional-derivative (PD) controller for tracking the trajectories and stabilizing the system, such as to remain in a region where its dynamics are well approximated by the planar modeling. To better account for the nonminimum phase balancing dynamics, its gains are partially selected from an LQR (see Section IV-B).
3) We discuss choices related to state-estimation, system identification, and hardware which were relevant for successful real-world deployment (see Section IV-C).

Like many related TO works, e.g., [7], [40], [41], [48], [49], we assume a map of the terrain to be given.[3] This leaves some open questions regarding the performance of our approach under less ideal conditions. As part of the simulations and experiments in Section V, we therefore provide a preliminary evaluation of robustness against faulty terrain mapping, which also emulates the case of contact timing mismatch. A more thorough evaluation of this aspect would require extensive additional testing and is thus left to future work. We conclude by discussing further promising lines of research and extensions in Section VI.

[3]The interested reader is referred to [68], [69] for two state of the art mapping approaches.

II. Modeling

As has been outlined in Section I-A, appropriate modeling of the system is crucial for computing solutions in reasonable time. Therefore, this section focuses on the formulation and parametrization of the dynamics equations in relation to the terrain, since they constitute main drivers of complexity.

The arguably most restrictive choice we make in our modeling is the treatment of the system as a planar one. The plane is chosen to be aligned with the direction of driving and balancing, as shown in Fig. 2. Loosely speaking, for typical (bipedal) wheeled–legged balancing robots this plane matches the main operating direction, constitutes a symmetry plane of the mechanical design, and represents the plane with the most challenging dynamics—they are unstable and nonminimum phase—to control. As we show in the experiments for the case of *Ascento*, the out-of-plane dynamics can be regulated well enough by a high-gain tracking controller. This makes the planar modeling a good approximation. We thus argue that prioritizing accuracy of the planar modeling over generality of the 3-D one is in many respects a sensible choice.

This choice drastically reduces complexity of the dynamics equations and constraints by removing, among others, gyroscopic effects, spatial friction, and the possibility of rank-deficient ground contacts. In return, the planar rigid-body dynamics can be modeled in their full nonlinearity without any simplifications and, even for long horizons, computation times of our TO scheme remain in the order of a few seconds.

The rest of this section details the planar modeling. First, we explain how we obtain contact phases and locations from a planar terrain cross section in Section II-A. Then, in Section II-B we show how to derive the rigid-body dynamics from a set of generalized coordinates where for the wheel a minimal, arc-length related parametrization is employed. Finally, we discuss how to account for constraints on the ground reaction forces and for impactive transitions by a reprojection from the constraint-consistent space, in Sections II-C and II-D, respectively.

A. Terrain Phase Generation

Let an upcoming planar terrain cross section be given and assume for the following derivation that contact with the ground is maintained at all times (we will weaken this assumption in Section III-H). We define an inertial frame I that has its z-axis pointing upwards w.r.t. gravity, see Fig. 2. Our approach starts by subdividing the terrain into—or approximating it by—an ordered set $\mathcal{T}$ of adjacent curve segments c_i which describe the height of the terrain as a function of the x-coordinate and which are assumed to be at least three times continuously differentiable

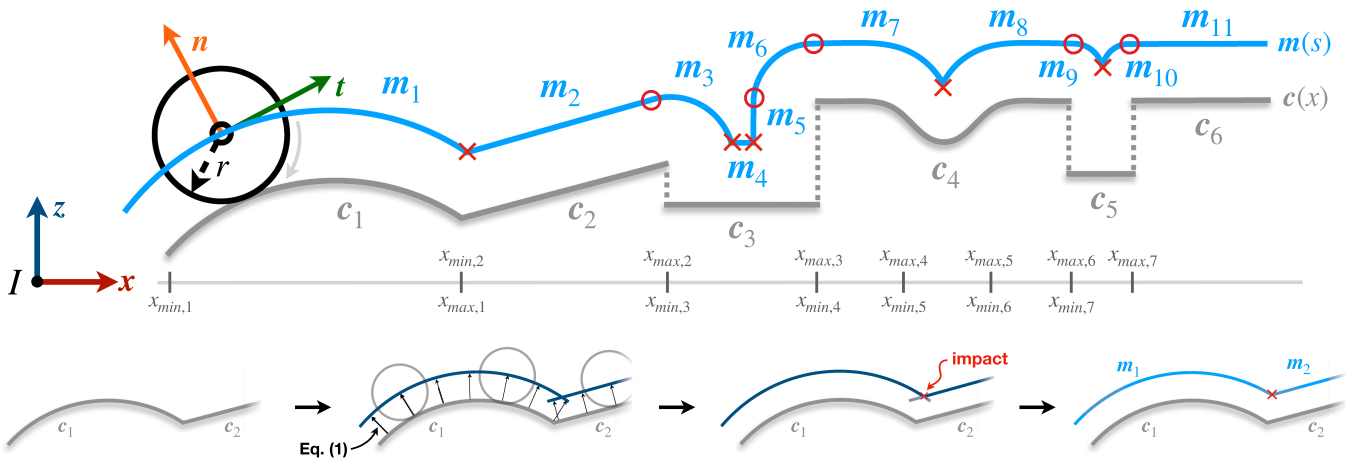


Fig. 2. *Top:* Example of a terrain composed of segments c_i and of the corresponding midpoint curve $\boldsymbol{m}(s)$. *Bottom:* Procedure of generating the continuous midpoint curve for the case of the first two terrain segments.

on their bounded domain, i.e.,

$$\mathcal{T} = \left\{ c_1, \ldots, c_{N_c} \,\middle|\, c_i(x) \in \mathcal{C}^3 \text{ for } x_{\min,i} \le x \le x_{\max,i} \right\}$$

where $[x_{\min,i},\, x_{\max,i}]$ is the domain of the ith curve segment. This allows to model almost any rigid terrain of interest, since it is only required that the number of discontinuities be finite. In particular, nonsmooth terrain including steps and stairs can be modeled.

Based on this subdivision we construct contact phases during which the dynamics of a wheeled system exhibits smooth properties. We do this by a procedure which, intuitively speaking, corresponds to rolling a wheel of radius r over the terrain and tracing its center $\boldsymbol{m}$, see Fig. 2. For a given terrain section, this amounts to applying the transformation

$$\boldsymbol{m}_i(x) = \begin{bmatrix} x \\ c_i(x) \end{bmatrix} + r \frac{1}{\sqrt{1 + c_i'^2(x)}} \begin{bmatrix} -c_i'(x) \\ 1 \end{bmatrix} \qquad (1)$$

where c_i' denotes the derivative of c_i w.r.t. x. However, the resulting curve segments $\boldsymbol{m}_i$ may be overlapping or disconnected in the x-coordinate. As can be seen in Fig. 2, certain segments are shortened (e.g., $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ at the point where they meet) and additional ones are introduced (e.g., $\boldsymbol{m}_3$) to obtain a connected midpoint curve. Therefore, the x-coordinate is unfit to parametrize the midpoint curve and we replace it by a closely related scalar parameter s. In some sense, s may be thought of as adding, shifting, distorting, and removing patches from the domain of x where appropriate. We will refer to s as *arc parameter*, since it is closely related to the arc-length of the curve $\boldsymbol{m}$. The reason for not simply using the arc-length, despite its direct physical interpretation, is that it would overly limit us in the type of segments that could be expressed in closed form, e.g., polynomials of order ≥ 2.

Let us now consider in more details the two problematic cases of intersecting and disconnected segments (the following examples provided in parentheses refer to Fig. 2).

1) In the case where two adjacent segments $\boldsymbol{m}_i$ and $\boldsymbol{m}_{i+1}$ intersect within the intersection of the *interior* of their domains (e.g., $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$), the domains must be shrunk accordingly. The point of intersection may either be determined analytically or by numerical computation using techniques such as gradient descent. Furthermore, we note that the resulting overall wheel center curve $\boldsymbol{m}$ may exhibit a kink at the point of intersection, which results in an instantaneous directional change of the wheel motion and thus an impact on the system.
2) Otherwise, and if the endpoints do not coincide (which corresponds to a discontinuity), at least one additional segment must be inserted. As illustrated by the example of a step, the wheel's midpoint traces a circular arc when it arrives at the ledge (consider, for instance, the transition from $\boldsymbol{c}_2$ to $\boldsymbol{c}_3$ and the corresponding segment $\boldsymbol{m}_3$). Two subcases can then be distinguished. Either, the height difference of the adjacent terrain segments is less than or equal to the wheel radius, in which case the introduction of the circular segment is sufficient (as for the transition from $\boldsymbol{c}_2$ to $\boldsymbol{c}_3$). Or, the height difference is larger than the wheel radius, in which case a second, vertical segment must be inserted (which is the case for the transition from $\boldsymbol{c}_3$ to $\boldsymbol{c}_4$, where both a vertical segment, $\boldsymbol{m}_5$, and a circular one, $\boldsymbol{m}_6$, are required). In both subcases, the treatment of the additional segments and the adjacent ones is as in case 1).

An additional complication arises when terrain portions have curvature equal or smaller than the wheel radius (as it occurs at the middle of $\boldsymbol{c}_4$). Then, the wheel cannot smoothly roll over the terrain and the corresponding midpoint curve must be divided into two parts, of which the connection is treated as in case 1). A similar case is when a small terrain portion is trapped between two corners that have distance less than the wheel radius such that it cannot be touched by the wheel (as for the terrain segment $\boldsymbol{c}_5$). We may then simply remove the corresponding terrain portion and proceed as in case 2).

An unavoidable consequence of our parametrization is that the time derivative of s may be discontinuous at the transition

between adjacent segments. We emphasize that this is independent of the terrain exhibiting discontinuities or kinks—it is a mathematical phenomenon. In the following we will refer to these discontinuities as *virtual impacts*, since they only affect the parametrization but not the system dynamics (see also the end of Section II-D).

To summarize, the procedure results in N_m planar parametric curves $\boldsymbol{m}_i$, each twice continuously differentiable over the bounded domain of definition, i.e., $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{N_m}\}$, where

$$\boldsymbol{m}_i(s) \in \mathcal{C}^2 \times \mathcal{C}^2 \mid s_{\min,i} \leq s \leq s_{\max,i},$$

which compose the curve $\boldsymbol{m}$. Within each of these contact phases, the dynamics of the system are smooth and therefore continuously differentiable.

The significance of our approach is that, by defining contact phases, the difficult combinatorial nature of the problem is reduced to the comparatively much simpler task of having to find the optimal durations of an ordered set of phases.

With regard to the TO we note that while our approach predetermines the order in which different segments of the terrain are traversed, it does not restrict the direction of motion *within* the segments. For wheeled balancing systems this is of particular importance because of their nonminimum phase dynamics, which, to move in one direction, may first require motion in the opposite one.

B. System Dynamics

Building upon the description of the terrain introduced in the previous section, we next derive the dynamics of a wheeled balancing robot moving on it. We assume that it is reasonable that the system in question be treated as a planar one that consists of a wheel and an arbitrary mechanism of rigid bodies attached to the wheel by a link, which we shall call *shank*. Since the contact interaction between wheel and ground plays a central role in our approach, we employ a wheel-centric parametrization, which leads to a favorable algebraic structure. Namely, we define the generalized coordinates of the system as

$$\boldsymbol{q}_f = \begin{bmatrix} x & z & \varphi & \theta & q_1 & \ldots & q_{N_q} \end{bmatrix}^\top$$

where x, z denote the wheel's Cartesian coordinates, φ the wheel angle, and θ the tilt angle of the shank, see Fig. 3. The upper structure may have arbitrary design—we only assume that the relations governing its motion w.r.t. the shank are known and that they be parametrized by a minimal set of generalized coordinates $q_j, j = 1, 2, \ldots, N_q$. For compactness we define the state vector $\boldsymbol{x}_f = [\boldsymbol{q}_f^\top \ \dot{\boldsymbol{q}}_f^\top]^\top$. We gather the inputs actuating the system in the vector $\boldsymbol{\tau} = [\tau_W \ \tau_1 \ \ldots \ \tau_{N_\tau}]^\top$, where τ_W denotes the wheel torque and $\tau_1, \ldots, \tau_{N_\tau}$ the actuation torques (or forces) of the remaining actuated DOFs.

By applying the formalism of choice, e.g., Lagrangian mechanics, the equations of motion (EOM) may be derived in the form

$$\boldsymbol{M}_f(\boldsymbol{q}_f)\,\ddot{\boldsymbol{q}}_f - \boldsymbol{h}_f\left(\boldsymbol{q}_f, \dot{\boldsymbol{q}}_f\right) = \boldsymbol{S}_f^\top \boldsymbol{\tau} + \boldsymbol{J}_C^\top \boldsymbol{\lambda}_C. \tag{2}$$

$\boldsymbol{M}_f$ denotes the symmetric, positive definite mass matrix and $\boldsymbol{h}_f$ collects remaining inertial forces, gravitational terms, and contributions stemming, for instance, from springs. The selection matrix $\boldsymbol{S}_f$ maps actuations to their respective DOF. Finally, $\boldsymbol{J}_C$ denotes the *contact Jacobian* and $\boldsymbol{\lambda}_C \in \mathbb{R}^2$ the ground contact forces, which we discuss in the following. From here onward, where it does not hinder comprehension, we omit listing arguments in favor of a more compact notation.

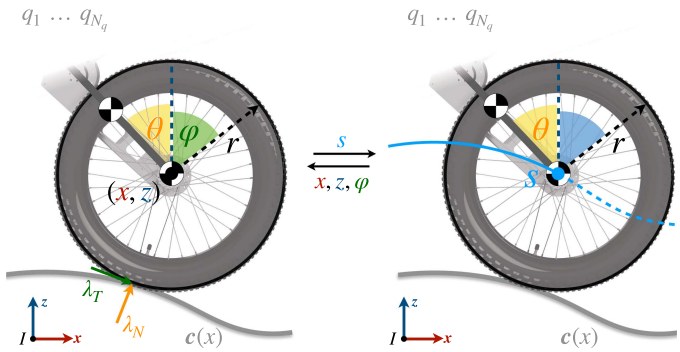


Fig. 3. Illustration of the proposed reparametrization. *Left:* The kinematically unconstrained model (formulated in the *full* space of generalized coordinates), which is dynamically constrained through explicit contact forces. *Right:* The kinematically inherently constraint consistent model (formulated using a *minimal* set of generalized coordinates), which therefore does not require explicit knowledge of the contact forces.

There exist two approaches for modeling the dynamics of a system which is in contact with a surface. One is based on a dynamic, the other on a kinematic treatment of the contact constraints. We employ the kinematic approach, which in our case consists of specifying the kinematics through a minimal representation which is inherently constraint-consistent. We provide the justification for this choice in the remark at the end of this section, along with further comments.

Let us assume perfect rolling, that is, no (unexpected) lift-off, no penetration, and no slip between wheel and ground takes place. The key observation is then that the wheel center curve $\boldsymbol{m}$ uniquely determines the wheel angle φ (up to its initial value). That is, we may parametrize the wheel's state—consisting of the angle and the x- and z-coordinates of the center—by the single arc parameter s. A minimal set of generalized coordinates is therefore given by

$$\boldsymbol{q}_s = \begin{bmatrix} s & \theta & q_1 & \ldots & q_{N_q} \end{bmatrix}^\top.$$

For compactness we also define the minimal state vector $\boldsymbol{x}_s = [\boldsymbol{q}_s^\top \ \dot{\boldsymbol{q}}_s^\top]^\top$.

We must next find the transformation mapping $\boldsymbol{q}_s$ to the generalized coordinates $\boldsymbol{q}_f$ and the transformations relating their first and second time derivatives. These transformations will allow us to project the EOM (2) to a constraint-consistent subspace. Finding these transformations amounts to deriving expressions for x, z, φ and their time derivatives as functions of s and its time derivatives (the transformations for the remaining generalized coordinates, θ, q_1,...,q_{N_q}, are the identity). Of these, the expressions for the wheel angle are the most involved, and we thus show how to bring them into a convenient form, which is also suitable for subsequent application of algorithmic differentiation.

We have that $[x \quad z]^\top = \boldsymbol{m}(s)$ and that $r\varphi$ equals the arc length traveled by the wheel's center, that is,

$$\varphi\, r = \int_0^s \|\boldsymbol{m}'(\sigma)\|\, d\sigma \tag{3}$$

where $\boldsymbol{m}'(s) = \frac{d}{ds}\boldsymbol{m}(s)$. We thus obtain the transformation mapping $\boldsymbol{q}_s$ to $\boldsymbol{q}_f$ as

$$\boldsymbol{f}_{\boldsymbol{q}_f}(\boldsymbol{q}_s) = \begin{bmatrix} \boldsymbol{m}(s)^\top & \frac{1}{r}\int_0^s \|\boldsymbol{m}'(\sigma)\|\, d\sigma & q_1 & \dots & q_{N_q} \end{bmatrix}^\top. \tag{4}$$

However, the arc length integral may not admit a closed-form solution. Fortunately, there is no need to evaluate it, since the dynamics of the system are invariant w.r.t. φ due to the rotational symmetry of the wheel. The invariance does not hold w.r.t. the time derivatives of φ, though.

The transformation mapping $\dot{\boldsymbol{q}}_s$ to $\dot{\boldsymbol{q}}_f$ is obtained from differentiation of $\boldsymbol{f}_{\boldsymbol{q}_f}(\boldsymbol{q}_s)$, that is,

$$\dot{\boldsymbol{q}}_f = \frac{d}{dt}\boldsymbol{f}_{\boldsymbol{q}_f}(\boldsymbol{q}_s) = \boldsymbol{Q}_f\, \dot{\boldsymbol{q}}_s$$
$$\text{where} \quad \boldsymbol{Q}_f = \frac{\partial}{\partial \boldsymbol{q}_s^\top}\boldsymbol{f}_{\boldsymbol{q}_f}(\boldsymbol{q}_s). \tag{5}$$

The entry in the Jacobian $\boldsymbol{Q}_f$ that corresponds to $\dot{\varphi}$ is computed by time differentiation of (3). Applying the Leibniz integral rule, we obtain

$$\dot{\varphi} = \frac{1}{r}\frac{d}{dt}\int_0^s \|\boldsymbol{m}'(\sigma)\|\, d\sigma = \frac{1}{r}\left(\|\boldsymbol{m}'(s)\|\ \dot{s} - \|\boldsymbol{m}'(0)\|\ 0 + \int_0^s \frac{\partial}{\partial t}\|\boldsymbol{m}'(\sigma)\|\, d\sigma\right) = \frac{1}{r}\|\boldsymbol{m}'(s)\|\ \dot{s}. \tag{6}$$

The transformation relating $\ddot{\boldsymbol{q}}_f$ to the minimal representation is the time derivative of (5), that is,

$$\ddot{\boldsymbol{q}}_f = \dot{\boldsymbol{Q}}_f\, \dot{\boldsymbol{q}}_s + \boldsymbol{Q}_f\, \ddot{\boldsymbol{q}}_s. \tag{7}$$

Hereby, the terms expressing $\ddot{\varphi}$ as a function of the arc parameter can be computed by time differentiation of (6) as

$$\ddot{\varphi} = \frac{1}{r}\frac{d}{dt}\left(\|\boldsymbol{m}'(s)\|\ \dot{s}\right) = \frac{1}{r}\left(\frac{\boldsymbol{m}''(s)^\top \boldsymbol{m}'(s)}{\|\boldsymbol{m}'(s)\|}\,\dot{s}^2 + \|\boldsymbol{m}'(s)\|\ \ddot{s}\right) \tag{8}$$

from which the corresponding entry in $\dot{\boldsymbol{Q}}_f$ is immediately deduced.

Given the Jacobian $\boldsymbol{Q}_f$ and its time derivative, the projection of the EOM (2) takes the form

$$\boldsymbol{M}_s\, \ddot{\boldsymbol{q}}_s - \boldsymbol{h}_s = \boldsymbol{S}_s^\top\, \boldsymbol{\tau}$$
$$\text{where} \quad \boldsymbol{M}_s = \boldsymbol{Q}_f^\top \boldsymbol{M}_f \boldsymbol{Q}_f, \quad \boldsymbol{S}_s^\top = \boldsymbol{Q}_f^\top \boldsymbol{S}_f^\top$$
$$\boldsymbol{h}_s = \boldsymbol{Q}_f^\top\left(\boldsymbol{h}_f - \boldsymbol{M}_f \dot{\boldsymbol{Q}}_f\, \dot{\boldsymbol{q}}_s\right). \tag{9}$$

Note in particular that, by the principle of virtual work, $\boldsymbol{Q}_f^\top \boldsymbol{J}_C^\top \boldsymbol{\lambda}_C = \boldsymbol{0}$, i.e., the contact forces vanish since the kinematics are constraint-consistent.

For use as an explicit constraint in the TO, we invert (9). Defining

$$\boldsymbol{f}_{\ddot{\boldsymbol{q}}_s}(\boldsymbol{q}_s,\, \dot{\boldsymbol{q}}_s,\, \boldsymbol{\tau}) = \boldsymbol{M}_s^{-1}\left(\boldsymbol{S}_s^\top\, \boldsymbol{\tau} + \boldsymbol{h}_s\right) \tag{10}$$

the time derivative of the minimal state vector is given as

$$\dot{\boldsymbol{x}}_s = \begin{bmatrix} \dot{\boldsymbol{q}}_s \\ \boldsymbol{f}_{\ddot{\boldsymbol{q}}_s}(\boldsymbol{q}_s,\, \dot{\boldsymbol{q}}_s,\, \boldsymbol{\tau}) \end{bmatrix}. \tag{11}$$

Remark: We now come back to the reason for choosing the kinematic approach. In the dynamic approach the constraint forces are solved for by imposing a zero velocity constraint at acceleration level [70]. This approach usually experiences significant constraint drift, which must be counteracted by methods such as Baumgarte's stabilization [71]. The resulting formulations are often stiff, potentially containing nonsmooth switches, which can pose problems for gradient-based TO. Even with small integration steps a certain amount of terrain penetration is unavoidable, which is exacerbated when different contacts are active in short succession, such as in the case of a step. Since penetration and constraint drift can significantly alter contact scheduling, we consider the dynamic approach unfavorable for our purposes.

Alternative options to using a minimal representation would be the explicit introduction of kinematic constraints or the (closely related) use of differential algebraic equation integration techniques. We refrain from the former since it leads to additional constraints in the optimization and from the latter in favor of reduced computational complexity.

Finally, we note that there is a strong connection to the treatment of loop-closure constraints in parallel kinematic systems, where both methods are used. However, the corresponding constraints are typically bilateral and are thus less involved. The interested reader is referred to [17] for the dynamic approach and to Section IV-A for the kinematic one.

C. Contact Force Constraints

The minimal coordinate formulation introduced in the previous section models the contact constraints as bilateral—thereby allowing for pulling forces—and implicitly treats the ground as having an arbitrarily large friction coefficient (these are the assumptions corresponding to perfect rolling). The contact forces $\boldsymbol{\lambda}_C$ must therefore be constrained accordingly by (unilateral) friction cone constraints in the TO, see Section III-D. However, through the minimal coordinate representation, access to these forces is lost and a reprojection into the space corresponding to the generalized coordinates $\boldsymbol{q}_f$ must be applied to recover them.

This is done by determining $\boldsymbol{q}_f$ and its derivatives through relations (4), (5), (7), and then solving for the contact forces by evaluating

$$\boldsymbol{\lambda}_C = \left(\boldsymbol{J}_C \boldsymbol{M}_f^{-1} \boldsymbol{J}_C^\top\right)^{-1}\left(\boldsymbol{J}_C \boldsymbol{M}_f^{-1}\left(\boldsymbol{h}_f - \boldsymbol{S}_f^\top\, \boldsymbol{\tau}\right) - \dot{\boldsymbol{J}}_C\, \dot{\boldsymbol{q}}_f\right).$$

The abovementioned expression is obtained by noting that, for perfect rolling, the velocity of the point C of the wheel currently in contact with the ground must be zero. That is, $\boldsymbol{v}_C = \boldsymbol{J}_C\, \dot{\boldsymbol{q}}_f = \boldsymbol{0}$, from which a constraint on acceleration level

can be obtained by time differentiation:

$$\dot{\boldsymbol{J}}_C\,\dot{\boldsymbol{q}}_f + \boldsymbol{J}_C\,\ddot{\boldsymbol{q}}_f = \boldsymbol{0}.$$

Combining this constraint with the EOM (2) gives the desired expression for $\boldsymbol{\lambda}_C$. We remark that using (4) and (5) it can be expressed in terms of the minimal representation.

For completeness, we note that the contact Jacobian is

$$\boldsymbol{J}_C = \begin{bmatrix} \mathbb{I} & -r\,\boldsymbol{t}(s) & \boldsymbol{0} \end{bmatrix}$$

where $\boldsymbol{t}(s)$ is the (normalized) tangent vector of the wheel at C and $\mathbb{I}$ denotes the identity matrix. The tangent and normal vectors of the wheel at C can be obtained from the wheel center curve as

$$\boldsymbol{t}(s) = \frac{\boldsymbol{m}'(s)}{\|\boldsymbol{m}'(s)\|}, \quad \boldsymbol{n}(s) = \frac{1}{\|\boldsymbol{m}'(s)\|}\begin{bmatrix} -m_2'(s) \\ m_1'(s) \end{bmatrix}$$

respectively, where $m_1(s)$ and $m_2(s)$ are the x and z components of $\boldsymbol{m}(s)$, respectively. Since we define these quantities relative to the wheel instead of the terrain, they are always well-defined, even when the terrain has kinks.

To formulate traction force constraints and nonnegativity constraints, it is convenient to express the contact forces in a frame C which is aligned with the tangent and normal vector:

$${}_C\boldsymbol{\lambda}_C = \boldsymbol{R}_{IC}^\top\,\boldsymbol{\lambda}_C, \quad \text{where} \quad \boldsymbol{R}_{IC} = \begin{bmatrix} \boldsymbol{t}(s) & \boldsymbol{n}(s) \end{bmatrix}.$$

We will refer to the tangential and normal components of ${}_C\boldsymbol{\lambda}_C$ as λ_T and λ_N, respectively, which will be useful later.

D. Impactive Phase Transitions

It is in fact straightforward to extend our modeling to handle impactive phase transitions, e.g., as they occur when jumping, which is discussed in Section III-H. To this end, we must relate the preimpact velocity $\dot{\boldsymbol{q}}_s^-$ and the postimpact velocity $\dot{\boldsymbol{q}}_s^+$—that is, the left- and right-hand limits of the velocity at the impact event. This is again done via a reprojection into the space corresponding to the generalized coordinates $\boldsymbol{q}_f$, where the relation between pre- and postimpact velocities is given as

$$\dot{\boldsymbol{q}}_f^+ = \boldsymbol{N}\,\dot{\boldsymbol{q}}_f^-, \qquad \text{where}$$

$$\boldsymbol{N} = \mathbb{I} - \boldsymbol{M}_f^{-1}\boldsymbol{J}_C^{+\top}\left(\boldsymbol{J}_C^+\boldsymbol{M}_f^{-1}\boldsymbol{J}_C^{+\top}\right)^{-1}\boldsymbol{J}_C^+. \tag{12}$$

This relation can be derived by noting that integrating the EOM (2) over a single time instance t_0 for the case of an impact gives

$$\int_{\{t_0\}} \left(\boldsymbol{M}_f\,\ddot{\boldsymbol{q}}_f - \boldsymbol{h}_f - \boldsymbol{S}_f^\top\boldsymbol{\tau} - \boldsymbol{J}_C^\top\boldsymbol{\lambda}_C\right) dt$$
$$= \boldsymbol{M}_f\left(\dot{\boldsymbol{q}}_f^+ - \dot{\boldsymbol{q}}_f^-\right) - \boldsymbol{J}_C^{+\top}\tilde{\boldsymbol{\lambda}}_C = \boldsymbol{0} \tag{13}$$

where $\tilde{\boldsymbol{\lambda}}_C$ is the impulsive contact force and $\boldsymbol{J}_C^+$ is the contact Jacobian corresponding to the postimpact terrain phase. Assuming perfect rolling after the impact, $\boldsymbol{J}_C^+\dot{\boldsymbol{q}}_f^+ = \boldsymbol{0}$, which can be used to eliminate $\tilde{\boldsymbol{\lambda}}_C$. Namely, if (13) is premultiplied by $\boldsymbol{M}_f^{-1}$ and then by $\boldsymbol{J}_C^+$, the postimpact velocity $\dot{\boldsymbol{q}}_f^+$ drops out and one can solve for $\tilde{\boldsymbol{\lambda}}_C$ as a function of $\dot{\boldsymbol{q}}_f^-$. The desired expression (12) then easily follows.

We note that in the case of nonimpactive transitions, including virtual impacts, as introduced in Section II-A, the projection (12) produces $\dot{\boldsymbol{q}}_f^+ = \dot{\boldsymbol{q}}_f^-$. In other words, (12) holds for every transition. However, the impact projector $\boldsymbol{N}$ is not simply the identity for nonimpactive transitions; rather, $\dot{\boldsymbol{q}}_f^-$ lies in the nullspace of $(\boldsymbol{N} - \mathbb{I})$.

An important consideration for numerical implementation of the impact constraints is that, as a function of $\dot{\boldsymbol{q}}_s$, they are rank-deficient. Namely, $\dot{x}$, $\dot{z}$, $\dot{\varphi}$ can all be expressed as functions of $\dot{s}$ (and s). Observing that the mapping (6) between $\dot{\varphi}$ and $\dot{s}$ is always well-defined, we may thus simply discard the parts of the constraints corresponding to $\dot{x}$ and $\dot{z}$. This may be interpreted as a back-projection to the space corresponding to $\dot{\boldsymbol{q}}_s$. Accordingly, a transformation mapping $\dot{\boldsymbol{q}}_f$ to $\dot{\boldsymbol{q}}_s$ is given by

$$\boldsymbol{f}_{\dot{\boldsymbol{q}}_s}(s, \dot{\boldsymbol{q}}_f) = \begin{bmatrix} 0 & 0 & \frac{r}{\|\boldsymbol{m}'(s)\|} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \mathbb{I} \end{bmatrix}\dot{\boldsymbol{q}}_f.$$

Hence, the impact equation for adjacent contact phases is

$$\dot{\boldsymbol{q}}_s^+ = \boldsymbol{f}_{\dot{\boldsymbol{q}}_s}(s^-, \boldsymbol{N}\boldsymbol{Q}_f\,\dot{\boldsymbol{q}}_s^-).$$

III. Trajectory Optimization

This section details the NLP formulation of our TO scheme. Its basic structure is outlined in Fig. 4.

We emphasize that it is cost function-free, with the only intended tuning parameter being the maximal total trajectory time ΔT_{max}, which serves to adjust the aggressiveness of the motions. However, including a cost function is straightforward, though it may (moderately) increase convergence times of the solver.[4] In this regard, the reader should also keep in mind that the formulation of nonconvex optimization problems may be considered dependent on the employed solver. Namely, the solver may be seen as internally modifying the formulation depending on the way it handles constraints. Using for instance an interior-point method, as we propose, results in the implicit introduction of a cost function that penalizes proximity to inequality constraint boundaries. We discuss the choice of the solver and optimization setup in Section V-A.

A. Optimization Variables

As we enforce a high number of pure state constraints, we choose a direct transcription approach, which simultaneously optimizes for states and inputs. We include the phase durations ΔT_i as optimization variables since predetermined durations will in general lead to suboptimal solutions that require unnecessary acceleration and deceleration of the system—or might even render an otherwise well-posed problem infeasible.

We discretize the state and input variables by dividing the duration ΔT_i of each phase into equally spaced intervals delimited by $N_{k_i} + 1$ nodes. The resulting twofold covering of the time

[4] Using an objective that trades-off effort versus time optimality, we noticed an increase of convergence times in the range of 20% to 50% for the case of the *Ascento* robot, with the setup described in Section V-A.

$$
\begin{aligned}
&\text{find } \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{N_m}\ \boldsymbol{U}_1, \ldots \boldsymbol{U}_{N_m}\ \Delta T_1, \ldots \Delta T_{N_m} && \text{(optimization variables—Section III-A)}\\
&\text{s.t.}\quad \boldsymbol{X}(0) = \boldsymbol{x}_0, \quad \boldsymbol{X}(T) = \boldsymbol{x}_T, \quad 0 \le \Delta T_i, \quad \textstyle\sum_{i=1}^{N_m} \Delta T_i \le \Delta T_{max} && \text{(boundary conditions—Section III-B)}\\
&\quad \text{for phase } i \in \{1, \ldots, N_m\}\\
&\qquad \dot{\boldsymbol{X}}_i = \boldsymbol{f}_{\dot{\boldsymbol{x}}_i}(\boldsymbol{X}_i, \boldsymbol{U}_i) && \text{(system dynamics—Section III-C)}\\
&\qquad \boldsymbol{\lambda}_{C,i}(\boldsymbol{X}_i, \boldsymbol{U}_i) \in \mathcal{F}_i(\boldsymbol{X}_i, \boldsymbol{U}_i, \mu) && \text{(contact forces—Section III-D)}\\
&\qquad \boldsymbol{X}_{min} \le \boldsymbol{X}_i \le \boldsymbol{X}_{max} && \text{(state bounds—Section III-E)}\\
&\qquad \boldsymbol{X}_{i+1}[1] = \boldsymbol{h}_i(\boldsymbol{X}_i[N_{k_i}]) && \text{(phase transitions—Section III-F)}\\
&\qquad -\boldsymbol{U}_{max} \le \boldsymbol{U}_i \le \boldsymbol{U}_{max}, \quad -\boldsymbol{g}_T(\boldsymbol{U}_i) \le \boldsymbol{g}_{S,i}(\boldsymbol{X}_i) \le \boldsymbol{g}_T(\boldsymbol{U}_i) && \text{(actuator limitations—Section III-G)}
\end{aligned}
$$

Fig. 4. Nonlinear program (NLP) formulation of our TO scheme. Matrix inequalities are to be interpreted elementwise. The precise meaning of the constraints, which in favor of compactness are written with slight abuse of notation, is explained in the sections indicated on the right.

instance at the end, respectively start, of adjacent phases permits for an explicit formulation of impactive transition conditions, see Section III-F. For each phase i we gather the state and input variables in two matrices

$$
\boldsymbol{X}_i = \begin{bmatrix} \boldsymbol{q}_{s,i}[1] & \ldots & \boldsymbol{q}_{s,i}[N_{k_i}] \\ \dot{\boldsymbol{q}}_{s,i}[1] & \ldots & \dot{\boldsymbol{q}}_{s,i}[N_{k_i}] \end{bmatrix}
$$

$$
\boldsymbol{U}_i = \begin{bmatrix} \boldsymbol{\tau}_i[1] & \ldots & \boldsymbol{\tau}_i[N_{k_i}] \end{bmatrix}.
$$

This allows for a compact formulation of constraints related to the system dynamics since these are invariant within a phase.

B. Boundary Conditions

We constrain the first state $\boldsymbol{X}(0)$ to match the initial state $\boldsymbol{x}_0$ of the system and the last state $\boldsymbol{X}(T)$ to match a final state $\boldsymbol{x}_T$. We note that it may be desirable to select a steady state solution for the final state. In this case it is sensible to add corresponding input constraints $\boldsymbol{U}(T) = \boldsymbol{u}_T$, where $\boldsymbol{u}_T$ is the steady-state input.[5] Furthermore, each phase duration ΔT_i should be positive and we require the total duration of the trajectory to be smaller than a fixed value ΔT_{max}, which acts as a tuning parameter.

C. System Dynamics

We incorporate the dynamics model derived in Section II-B by a custom augmentation of the Hermite–Simpson collocation scheme, which we rely on because it represents a "sweet spot" regarding accuracy versus the number of function evaluations [72]. In the standard Hermite–Simpson method the state is approximated by cubic polynomials and the input by linear functions. It leads to third-order accuracy with only two evaluations of the dynamics per discretization interval. We sketch in the following how to extend the method to yield even better accuracy per function evaluation. The details are provided in Appendix A.

[5]The initial state may be a steady-state solution as well, e.g., when the system is starting from a steady-state balancing configuration. In either case, equilibrium points of the dynamics equations can be determined by nonlinear root finding.

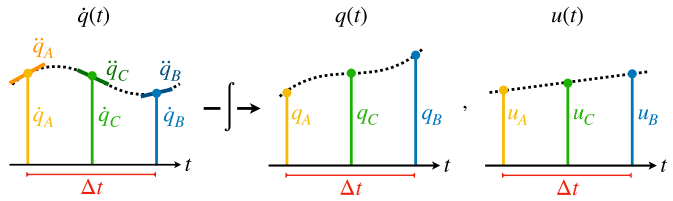


Fig. 5. Illustration of our augmented collocation scheme. Quantities are shown at the node points A and B and at the collocation point C. The cubic polynomials interpolating the generalized velocities (*left*) are integrated to quartic polynomials for interpolation of the generalized coordinates (*center*). Inputs are interpolated linearly (*right*).

Applying the Hermite–Simpson collocation scheme to a second-order dynamical system (implicitly) requires two sets of cubic polynomials—one for interpolation of the generalized coordinates and one for interpolation of the generalized velocities. This, however, leads to kinematic inconsistency between the two, even if the generalized velocities are simply the time derivatives of the generalized coordinates—as in our case. We resolve this issue by interpolating the generalized coordinates by quartic polynomials, which are obtained from integration of the cubic polynomials used for interpolation of the generalized velocities. The result is fourth-order accuracy in the generalized coordinates, as the derivatives of the coordinate polynomials match the velocity polynomials, which provide third-order accuracy. As we show, this does not require any additional function evaluations per discretization interval.

Given two adjacent node points A and B, with a central collocation point C—see Fig. 5—the extended scheme takes the form shown in the following. We omit phase and parametrization specific indices since the statements hold in general. First, the accelerations at the node points are expressed through the dynamics [see (10)] as

$$
\ddot{\boldsymbol{q}}_A = \boldsymbol{f}_{\ddot{\boldsymbol{q}}}(\boldsymbol{q}_A, \dot{\boldsymbol{q}}_A, \boldsymbol{\tau}_A), \quad \ddot{\boldsymbol{q}}_B = \boldsymbol{f}_{\ddot{\boldsymbol{q}}}(\boldsymbol{q}_B, \dot{\boldsymbol{q}}_B, \boldsymbol{\tau}_B).
$$

Then, the quantities at the central collocation point are obtained through interpolation by the formulae (see Appendix A for the

derivation)

$$\boldsymbol{q}_C = \frac{\Delta t}{32}\left(13\,\dot{\boldsymbol{q}}_A + 3\,\dot{\boldsymbol{q}}_B\right) + \frac{\Delta t^2}{192}\left(11\,\ddot{\boldsymbol{q}}_A - 5\,\ddot{\boldsymbol{q}}_B\right) + \boldsymbol{q}_A \tag{14}$$

$$\dot{\boldsymbol{q}}_C = \frac{1}{2}\left(\dot{\boldsymbol{q}}_A + \dot{\boldsymbol{q}}_B\right) + \frac{\Delta t}{8}\left(\ddot{\boldsymbol{q}}_A - \ddot{\boldsymbol{q}}_B\right) \tag{15}$$

$$\ddot{\boldsymbol{q}}_C = \frac{-3}{2\,\Delta t}\left(\dot{\boldsymbol{q}}_A - \dot{\boldsymbol{q}}_B\right) - \frac{1}{4}\left(\ddot{\boldsymbol{q}}_A + \ddot{\boldsymbol{q}}_B\right) \tag{16}$$

$$\boldsymbol{\tau}_C = \frac{1}{2}\left(\boldsymbol{\tau}_A + \boldsymbol{\tau}_B\right)$$

where Δt is the discretization interval obtained by dividing the current phase duration ΔT into N_k equally spaced collocation intervals. Finally, these expressions are combined into the two constraints

$$\ddot{\boldsymbol{q}}_C = \boldsymbol{f}_{\dot{\boldsymbol{q}}}\left(\boldsymbol{q}_C,\, \dot{\boldsymbol{q}}_C,\, \boldsymbol{\tau}_C\right) \tag{17}$$

$$\boldsymbol{q}_B = \frac{\Delta t}{2}\left(\dot{\boldsymbol{q}}_A + \dot{\boldsymbol{q}}_B\right) + \frac{\Delta t^2}{12}\left(\ddot{\boldsymbol{q}}_A - \ddot{\boldsymbol{q}}_B\right) + \boldsymbol{q}_A \tag{18}$$

which are a hybrid of a differential—the first constraint—and an integral method—the second one.[6] For an entire phase, these constraints may be gathered (under slight bending of notation) as $\dot{\boldsymbol{X}}_i = \boldsymbol{f}_{\dot{\boldsymbol{x}}_i}(\boldsymbol{X}_i, \boldsymbol{U}_i)$.

D. Friction Cone

As discussed in Section II-C, the dynamics modeling from Section II-B must be complemented by friction cone constraints. Letting μ denote the coefficient of static friction, these are given as

$$\lambda_{N_i}(\boldsymbol{x}_{s,i}, \boldsymbol{\tau}_i) \geq \boldsymbol{0}, \quad |\lambda_{T_i}(\boldsymbol{x}_{s,i}, \boldsymbol{\tau}_i)| \leq \mu\, \lambda_{N_i}(\boldsymbol{x}_{s,i}, \boldsymbol{\tau}_i).$$

For an entire phase, these may be written (under slight abuse of notation) as the set constraint

$$\boldsymbol{\lambda}_{C,i}(\boldsymbol{X}_i, \boldsymbol{U}_i) \in \mathcal{F}_i(\boldsymbol{X}_i, \boldsymbol{U}_i, \mu).$$

E. State Bounds

Within a given phase, the arc parameter s must be constrained to stay between the corresponding start and end values, i.e., $s_{i,\min} \leq s_i \leq s_{i,\max}$.[7]

Joint position limits are respected by constraints of the form $q_{j,\min} \leq q_j \leq q_{j,\max}$. Joint velocity limits could be introduced as well, but are preferably modeled as torque-dependent constraints, which more accurately describe the behaviour of typical actuators, see Section III-G.

Furthermore, it is necessary to enforce tilt angle bounds $\theta_{\min} \leq \theta \leq \theta_{\max}$ to keep the system in an upright balancing configuration. We note that, by omitting these bounds during flight phases (see Section III-H) and by imposing a constraint on the tilt angle to change by 2π, trajectories containing flips can be generated.

Finally, we experienced that for fast dynamic motions, such as jumping (see Section III-H), it may be required to enforce some of the abovementioned state bounds at increased rates. This can be done without a finer discretization by imposing them not only at the node points but also at the central collocation points.

F. Phase Transitions

At phase transitions, the coordinates $\boldsymbol{q}_s$ must remain continuous, that is, $\boldsymbol{q}_{s,i+1}[1] = \boldsymbol{q}_{s,i}[N_{k_i}]$. As follows from the discussion at the end of Section II-D the velocities $\dot{\boldsymbol{q}}_s$, which may be discontinuous, must meet the condition

$$\dot{\boldsymbol{q}}_{s,i+1}[1] = \boldsymbol{f}_{\dot{\boldsymbol{q}}_s}\left(s_i[N_{k_i}], \boldsymbol{N}\,\boldsymbol{Q}_f\,\dot{\boldsymbol{q}}_{s,i}[N_{k_i}]\right).$$

These two constraints may be combined as

$$\begin{aligned}\boldsymbol{x}_{s,i+1}[1] &= \boldsymbol{h}_i(\boldsymbol{x}_{s,i}[N_{k_i}]) \\ &= \begin{bmatrix} \boldsymbol{q}_{s,i}[N_{k_i}] \\ \boldsymbol{f}_{\dot{\boldsymbol{q}}_s}\left(s_i[N_{k_i}], \boldsymbol{N}\,\boldsymbol{Q}_f\,\dot{\boldsymbol{q}}_{s,i}[N_{k_i}]\right) \end{bmatrix}.\end{aligned} \tag{19}$$

We have found it useful to also enforce torque continuity at transitions through constraints $\boldsymbol{U}_i[N_{k_i}] = \boldsymbol{U}_{i+1}[1]$ in order to prevent undesirably high jerk caused by jumps in the torques. However, these constraints are not strictly necessary because the dynamics of current-controlled actuators are typically very fast compared to the ones of mechanical systems.

G. Actuator Limitations

We account for actuator torque limitations by constraints

$$-\boldsymbol{U}_{\max} \leq \boldsymbol{U}_i \leq \boldsymbol{U}_{\max}$$

and treat actuator velocity limits as torque dependent—as done in [73]. Approximating the relation between maximal joint velocity and torque to be linear affine should typically be sufficient. We note that relation (6) must be used to obtain $\dot{\varphi}$ and that the shank's tilt velocity $\dot{\theta}$ must be accounted for. The proposed constraints are of the form

$$-\boldsymbol{v}_{\max} + \boldsymbol{C}\,\boldsymbol{\tau}_i[k] \leq \boldsymbol{g}_{S,i}(\boldsymbol{x}_s) \leq \boldsymbol{v}_{\max} - \boldsymbol{C}\,\boldsymbol{\tau}_i[k]$$

for some constant diagonal matrix $\boldsymbol{C}$, vector $\boldsymbol{v}_{\max}$, and

$$\boldsymbol{g}_{S,i}(\boldsymbol{x}_s) = \begin{bmatrix} \frac{1}{r}\left\|\boldsymbol{m}_i'(s)\right\|\dot{s} + \dot{\theta} & \dot{q}_1 & \dots & \dot{q}_{N_q} \end{bmatrix}^\top.$$

For an entire phase, the actuator velocity constraints may be gathered (under slight abuse of notation) as

$$-\boldsymbol{g}_T(\boldsymbol{U}_i) \leq \boldsymbol{g}_{S,i}(\boldsymbol{X}_i) \leq \boldsymbol{g}_T(\boldsymbol{U}_i).$$

H. Extension to Jumping Maneuvers

It is straightforward to extend the proposed TO scheme to handle flight phases, that is, phases in which the system is not in contact with the ground. During such phases the state vector

[6] The attentive reader might have counted a total of three function evaluations for a single collocation interval. However, as in the standard Hermite–Simpson method, the evaluation of the dynamics at node B can be reused for the next time interval.

[7] For the first and the last phase we have found it useful to enlarge the boundaries slightly beyond the values of s corresponding to $\boldsymbol{x}_0$ and $\boldsymbol{x}_T$, in order to account for the nonminimum phase balancing dynamics. Extensions around 20%–80% of the wheel radius are in our experience sufficient and lead to notably smoother motions.

$\boldsymbol{x}_f$ is used in place of $\boldsymbol{x}_s$, that is,

$$\boldsymbol{X}_{i=\text{flight}} = \begin{bmatrix} \boldsymbol{q}_{f,i}[1] & \dots & \boldsymbol{q}_{f,i}[N_{k_i}] \\ \dot{\boldsymbol{q}}_{f,i}[1] & \dots & \dot{\boldsymbol{q}}_{f,i}[N_{k_i}] \end{bmatrix}.$$

Accordingly, the dynamics (2), with $\boldsymbol{\lambda}_C$ set to zero, are used in the collocation scheme presented in Section III-C and the friction cone constraints drop out.

Through the parametrization in $\boldsymbol{q}_f$ it is possible to directly specify constraints on the Cartesian wheel center coordinates, such as bounds for a minimal jump height.

Between adjacent flight phases the state remains continuous and the transition condition thus becomes $\boldsymbol{x}_{f,i+1}[1] = \boldsymbol{x}_{f,i}[N_{k_i}]$. During takeoff, i.e., transitions from a contact phase to a flight phase, we have continuity of the state as well:

$$\boldsymbol{x}_{f,i+1}[1] = \begin{bmatrix} \boldsymbol{f}_{\boldsymbol{q}_f}(\boldsymbol{q}_s[N_{k_i}]) \\ \boldsymbol{Q}_f\, \dot{\boldsymbol{q}}_{s,i}[N_{k_i}] \end{bmatrix}.$$

We note that due to invariance of the dynamics w.r.t. φ, the constraint on φ may be omitted, see Section II-B. During landing an impactive transition occurs and the transition condition is thus given as

$$\begin{bmatrix} \boldsymbol{f}_{\boldsymbol{q}_f}(\boldsymbol{q}_{s,i+1}[1]) \\ \dot{\boldsymbol{q}}_{s,i+1}[1] \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_{f,i}[N_{k_i}] \\ \boldsymbol{f}_{\dot{\boldsymbol{q}}_s}\left(\boldsymbol{N}\, \dot{\boldsymbol{q}}_{f,i}[N_{k_i}]\right) \end{bmatrix}.$$

To conclude, we note that there is no constraint that would enforce the takeoff and landing to occur at the segment boundaries, thus leaving the optimization the freedom to choose the optimal transition locations. All that is required is the addition of a phase with the abovementioned parametrization.

IV. Application to the Ascento Robot

In this section we discuss the application of the proposed TO scheme to *Ascento*—a wheeled bipedal balancing robot with a total of four actuated DOFs. For each leg one motor drives the wheel and another the knee, which moreover is supported by a spring. The design of the legs is symmetric and their linkages only permit motion parallel to the symmetry plane, which makes the planar modeling of Section II-B a reasonable choice. The reader should not be deceived by the planar model having a total of five DOFs only—some considerable complexity is concealed in the closed-loop four-bar linkage. That we may model it without resorting to crude simplifications demonstrates one of the strengths of our approach.

A. Robot Model

We model the four-bar linkage in a kinematically constraint-consistent way. This is in contrast to our previous work [17], which followed a dynamical approach. The reader is referred to Section II-B for the merits of the kinematic approach over the dynamic one. We furthermore remark that the modeling presented in the following was also used in our recently published work treating the design optimization of *Ascento's* four-bar linkage [74]. However, neither the derivation nor the resulting equations were shown therein.

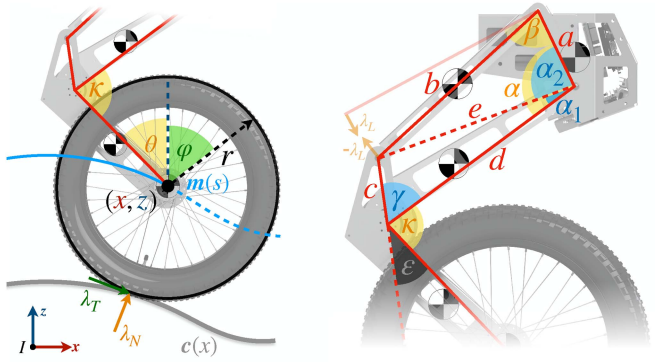


Fig. 6. Parametrization of the planar *Ascento* robot model for the wheel (*left*) and the closed-loop four-bar linkage (*right*).

Following the modeling procedure of Section II-B, we choose for the wheel-centric parametrization the set of generalized coordinates $\boldsymbol{q}_f = [x\ z\ \varphi\ \theta\ \kappa]^\top$, where κ is the knee angle, see Fig. 6. Other choices are possible for parametrizing the four-bar linkage, but this one results in a favorable structure of the EOM because the spring and the motor directly act on the knee. The actuation vector is given by $\boldsymbol{\tau} = [\tau_W\ \ \tau_K]^\top$, where τ_W and τ_K denote the wheel and knee actuation torques, respectively.

Next, we open the loop. To parametrize the opened linkage we define the augmented state

$$\boldsymbol{q}_a = \begin{bmatrix} x & z & \varphi & \theta & \kappa & \alpha & \beta \end{bmatrix}^\top$$

where α and β are auxiliary variables (other choices are possible), see Fig. 6. Then, by applying the formalism of choice, the EOM of the augmented system can be derived in the form

$$\boldsymbol{M}_a(\boldsymbol{q}_a)\ \ddot{\boldsymbol{q}}_a - \boldsymbol{h}_a(\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a) = \boldsymbol{S_a}^\top \boldsymbol{\tau} + \boldsymbol{J}_L^\top \boldsymbol{\lambda}_L \tag{20}$$

where $\boldsymbol{J}_L$ the denotes the contact Jacobian of the loop and $\boldsymbol{\lambda}_L \in \mathbb{R}^2$ the forces required to keep the loop closed. Paralleling the development of Section II-B, $\boldsymbol{\lambda}_L$ can be eliminated by projecting the EOM to a constraint-consistent subspace. Accordingly, the auxiliary states α and β must be expressed as functions of κ. Through repeated application of the cosine-rule one finds that

$$\begin{aligned} \alpha(\kappa) &= \alpha_1(\kappa) + \alpha_2(\kappa), \\ \beta(\kappa) &= \arccos\left(\frac{a^2 + b^2 - e^2(\kappa)}{2\,a\,b}\right) \\ \text{where}\quad \gamma(\kappa) &= \pi - \kappa - \varepsilon \\ e(\kappa) &= \sqrt{c^2 + d^2 - 2\,c\,d\cos(\gamma(\kappa))} \\ \alpha_1(\kappa) &= \arccos\left(\frac{d^2 + e^2(\kappa) - c^2}{2\,d\,e(\kappa)}\right) \\ \alpha_2(\kappa) &= \arccos\left(\frac{a^2 + e^2(\kappa) - b^2}{2\,a\,e(\kappa)}\right). \end{aligned}$$

It can be shown that the functions $\alpha(\kappa)$ and $\beta(\kappa)$ are smoothly differentiable in κ over the entire leg linkage operating range.

From these relations it is straightforward to derive the differential relation $\dot{\boldsymbol{q}}_a = \boldsymbol{Q}_a\, \dot{\boldsymbol{q}}_f$ and the derivative $\dot{\boldsymbol{Q}}_a$ required to

form the constraint consistent EOM

$$\boldsymbol{M}_f\,\ddot{\boldsymbol{q}}_f - \boldsymbol{h}_f = \boldsymbol{S}_f^\top\,\boldsymbol{\tau}$$
$$\text{where}\quad \boldsymbol{M}_f = \boldsymbol{Q}_a^\top \boldsymbol{M}_a \boldsymbol{Q}_f,\quad \boldsymbol{S}_f^\top = \boldsymbol{Q}_a^\top \boldsymbol{S}_a^\top$$
$$\boldsymbol{h}_f = \boldsymbol{Q}_a^\top \left(\boldsymbol{h}_a - \boldsymbol{M}_a \dot{\boldsymbol{Q}}_a\,\dot{\boldsymbol{q}}_f\right).$$

Note that, by the principle of virtual work, $\boldsymbol{Q}_a^\top \boldsymbol{J}_L^\top \boldsymbol{\lambda}_L = \boldsymbol{0}$, i.e., the loop closure forces vanish since the kinematics are constraint-consistent.

From this point onward, the modeling is identical to Section II-B, with the set of minimal coordinates given by $\boldsymbol{q}_s = [s\ \ \theta\ \ \kappa]^\top$.

B. Tracking Control

We propose a high-gain PD controller for tracking the 2-D trajectories and stabilizing the full 3-D system while keeping it within a region where its dynamics are well approximated by the planar modeling used in the TO. The overall controller, which is explained in the following, takes the form

$$\begin{bmatrix} \tau_{W_l} - \tau_{W_r} \\ \tau_{K_l} - \tau_{K_r} \\ \tau_{W_l} + \tau_{W_r} \\ \tau_{K_l} + \tau_{K_r} \end{bmatrix} = \begin{bmatrix} k_{P,\psi}\,(\psi^* - \psi) + k_{D,\psi}\,(\dot{\psi}^* - \dot{\psi}) \\ k_{P,\phi}\,(\phi^* - \phi) + k_{D,\phi}\,(\dot{\phi}^* - \dot{\phi}) \\ k_{P,\chi}\,(\chi^* - \chi) + k_{D,\chi}\,(\dot{\chi}^* - \dot{\chi}) \\ 0 \end{bmatrix}$$
$$+ \begin{bmatrix} 0 \\ 0 \\ \bar{\tau}_W + k_{P,\varphi}\,(\bar{\varphi}^* - \bar{\varphi}) + k_{D,\varphi}\,(\dot{\bar{\varphi}}^* - \dot{\bar{\varphi}}) \\ \bar{\tau}_K + k_{P,\kappa}\,(\bar{\kappa}^* - \bar{\kappa}) + k_{D,\kappa}\,(\dot{\bar{\kappa}}^* - \dot{\bar{\kappa}}) \end{bmatrix} \tag{21}$$

where the subscripts l and r denote quantities associated to the left and right leg, respectively, the superscript $*$ denotes desired quantities (distinguishing them from current/measured ones), and $\bar{\varphi} = \frac{1}{2}(\varphi_l + \varphi_r)$ and $\bar{\kappa} = \frac{1}{2}(\kappa_l + \kappa_r)$ denote the average wheel and knee angles, respectively.[8] Furthermore, ψ, ϕ, and χ denote the yaw, roll, and pitch angle of the robot's base. The desired quantities $\bar{\varphi}^*$, $\dot{\bar{\varphi}}^*$, $\bar{\kappa}^*$, $\dot{\bar{\kappa}}^*$, $\bar{\theta}^*$, $\dot{\bar{\theta}}^*$, and the feedforward torques $\bar{\tau}_W$, $\bar{\tau}_K$ are chosen as the solutions of the TO for the current time instance. We remark that this requires locating the node points in the TO solution corresponding to the current time instance. The polynomials employed by the collocation scheme are given in Hermite form by these nodes and represent a natural choice for interpolation.

For the case of *Ascento* it is practical to describe the orientation of the base by an intrinsic yaw–pitch–roll—that is, z-y'-x"—sequence of Euler angles. The reason is that for this sequence changes of the orientation can be related to the actuation through simple linear relations, as indicated in the control law—yaw motion is achieved by differential wheel torques, roll motion by differential knee torques, and pitch motion by coinciding wheel torques.

To comply with the planar modeling in the TO, the desired base roll and yaw angles and their rates are set to zero. The desired shank pitch angle and the desired average knee angle determine the desired base pitch angle through the relation

$$\chi^* = \theta^* + \bar{\kappa}^* - \alpha(\bar{\kappa}^*). \tag{22}$$

Time differentiation of this relation gives $\dot{\chi}^*$.

The gains $k_{P,\varphi}$, $k_{D,\varphi}$, $k_{P,\chi}$, and $k_{D,\chi}$ are chosen from an LQR. Its purpose is to stabilize the nonminimum phase balancing dynamics, which are primarily regulated by the wheel motion. Viewing the system as an inverted pendulum one sees that changes of the knee angle are somewhat secondary, since they relate to changes of the pendulum's length. The corresponding changes of the system's inertia may for several reasons be undesirable, and we thus assume the knee angle fixed in the LQR, which amounts to excluding its dynamics. The linear model used for the computation of the LQR is obtained as follows. First, the planar model (11) is linearized around the state given by the TO for the current time instance. We note that compared to linearizing around the mean of the measured left and right leg's state, this introduces no noise and has the advantage that s must not be estimated. Then, relations (6), (8), and (22) are used to transform the linearized model from one in the arc parameter s and shank tilt angle θ to one in the wheel angle φ and base pitch angle χ. Thus, the LQR is calculated for the subsystem related to $[\chi\ \ \dot{\chi}\ \ \varphi\ \ \dot{\varphi}]$. During flight phases, the portions of the controller related to the LQR are turned off, since no balancing takes place and the model (11) is not valid.[9]

Instead of tracking trajectories generated by the TO, the proposed controller may also be used to regulate the system toward a steady-state reference. Given a desired set-point, such as a desired base height and forward velocity, the planar model (11) can be solved through nonlinear root-finding for the corresponding equilibrium state and torques. These then replace the quantities previously assumed given by the TO.

We note that tuning the gains and the LQR weights is relatively straightforward since, for normal mode of operation, they are sufficiently decoupled. As a practical remark, we found it important to perform experiments for varying robot heights, such as to select parameters that work over the entire operating range of the knee angles.

To conclude, we wish to emphasize that much more sophisticated tracking controllers could be synthesized, which explicitly account for time mismatch between planned and executed mode switches. The design of such controllers is not trivial, however, and still an active field of research [75], [76], often requiring advanced state estimation approaches as well. Moreover, it is not clear whether such a controller is actually required in our case. In fact, as will be seen in Section V (see in particular the robustness evaluation), our experiments indicate that running the TO in receding horizon fashion might completely eliminate

[8] Instead of choosing the average for projection onto the plane, one could rely on more sophisticated model order reduction techniques. These are more involved, however, and did not seem necessary.

[9] We remark that using the wheels as flywheels to stabilize the system during flight phases is impractical for the purpose of tracking control. Not only would they have to spin at high rates but, moreover, decelerate to a specific, comparatively low angular velocity before transition to the next contact phase. This is less of an issue in the TO, however, since it accounts both for actuator limitations and the angular velocity required when landing.

the need for a more sophisticated tracking controller. In this regard we also want to mention that the present controller is considerably simpler than the hierarchical whole-body controller from our previous work [17], which was intended for overcoming unmodeled rough terrain. Nevertheless—even when tracking a simple steady-state solution, as described in Section V-D—the controller proposed in this work leads to superior robustness of the system when traversing unmodeled unstructured terrain. We attribute this to insufficient contact estimation quality, which would be required for the former approach to perform to its full extent.

C. Remarks on State Estimation and System Identification

The performance of any model-based control scheme is tied to the quality of the state estimation and the system identification, which in a work including real-world experiments should therefore at least be touched upon. We therefore provide a few remarks on these subjects in the following.

The state estimation employed in the real-world experiments presented in Section V is purely proprioceptive. The attitude and angular velocity of the robot's base are estimated by the extended Kalman filter of its inertial measurement unit. The joint angles are estimated by individual Kalman filters on acceleration level, akin to [77]. As prior the full planar dynamics model is used. That is, for a single joint angle q we choose as model for the Kalman filter

$$\begin{bmatrix} q[k+1] \\ \dot{q}[k+1] \\ \ddot{q}[k+1] \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q[k] \\ \dot{q}[k] \\ \ddot{q}[k] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ f_{\ddot{q}}(\boldsymbol{x}[k],\, \boldsymbol{u}[k]) \end{bmatrix}$$

$$q_{\text{meas}}[k] = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} q[k] & \dot{q}[k] & \ddot{q}[k] \end{bmatrix}^\top$$

where $q_{\text{meas}}[k]$ denotes the joint angle measured by the absolute motor encoder. Compared to a (weighted) moving-average difference scheme of similar smoothness—say, a tenth order filter in the estimation and tracking control loop, which runs at 400 Hz—the Kalman filter introduces 20 ms less delay on the estimated joint velocity. We found this reduction to be rather important, since it allowed for significantly more aggressive gains in the tracking controller.

To identify the inertial parameters, we meticulously weighed the components during assembly and relied heavily on the system's computer-aided design (CAD) model. In particular, we also account for the inertias of the motors' stators in the modeling. Nevertheless, we found the system to exhibit nonnegligible steady-state tracking errors. As an example, when commanded to balance at rest, it would roll away by up to half a meter from the expected position (until reaching the tilt angle offset permitting it to remain at rest). We attribute this effect mainly to uncertainty in the center of mass (COM) location of the base. It contains a multitude of electrical components, of which the inertial parameters could only be determined approximately. Indeed, a nonlinear least squares estimation[10] indicated the COM location to slightly differ (around 3 cm) from the CAD-based estimation. Accounting for this offset considerably reduced the steady-state tracking errors—for the abovementioned example to a few cm.

In all versions of the *Ascento* robot, the knee motor is supported by a custom, optimized spring system, which serves to approximately compensate the weight of the robot and thereby reduce power consumption. We model the torque τ_S generated by the spring system through a (smoothly differentiable) linear-quadratic model

$$\tau_S(\kappa) = k_1\,(\kappa - k_0) + k_2\,(\kappa - k_0)^2$$

of which we fitted the parameters k_0, k_1, k_2 by nonlinear least squares.[11]

The mechanical design of our first two prototypes were plagued by significant, partially direction-dependent friction in the wheel and knee joints, which considerably deteriorated the quality of the trajectory tracking. We found that approximating the friction by simple functions based on the hyperbolic tangent and accordingly modulating the torques output by the tracking controller (21) reduced the effects to a, more or less, acceptable amount. However, we do not provide the details, since attacking the problem at its root—the mechanical design—turned out to be much more effective. Namely, increasing rigidity and improving the suspension of the joint axes reduced friction to an amount where it barely affects the tracking and does not require modeling.

Finally, we found that deflating the tires—which is common practice in other applications when bumpy areas must be traversed—considerably helps trajectory tracking over terrain such as stairs. We attribute this to two effects. First, the increased damping reduces kickback—which is not modeled by our phase-based approach—e.g., when the wheel collides with the ledge of a step. We note that our prespecified phase ordering actually amounts to assuming the coefficient of restitution to be zero. Second, deflating increases the grip of the tires, which aids against slipping, for instance when the wheel has only contact to the ledge while driving up a step. Close observation revealed that the resulting grip could in fact be so high that over short amounts of time the system would behave as if the coefficient of static friction was higher than one. Accordingly, we found the commonly used choice $\mu = 0.8$ to be too conservative and used values up to $\mu = 1.0$.

[10] Viewing the system as a planar wheeled inverted pendulum, one sees that balancing at rest requires the x-component of the overall COM of the system—which depends on the COM location of the base—to lie above the wheel center. It is straightforward to exploit this observation for setting up a nonlinear least squares scheme which, based on the planar modeling, estimates the x and z components of the base COM from measurements of the base pitch angle χ and the knee angle κ. We generated corresponding datasets by commanding the system to balance at rest for 20 values of the knee angle, which were evenly spaced over its operating range.

[11] Similarly to the COM estimation of the base, we generated datasets by commanding the system to balance at rest for 20 values of the knee angle, which were evenly spaced over its operating range. We note that it was not possible to generate data by suspending the robot in the air and measuring the knee torques required to retract the legs, because the motors of the *Ascento* robots are not strong enough to fully counteract the springs when gravity is not compressing them through the weight of the base.

V. Results and Discussion

In this section, we discuss the implementation of our TO scheme, investigate some of its properties through simulation results, and validate the overall approach in real-world experiments. As the reader proceeds, they are advised to watch the accompanying video.[12] It includes all motions shown in the following, and additional ones.

A. Optimization Setup

Several aspects of our implementation, including the choice of the solver, turned out to be substantial for computational performance. We describe them in the following.

1) Problem Specification: We implemented our TO scheme using *CasADi* [78] through its MATLAB interface, whereby we heavily relied on *CasADi*'s autodifferentiation features—among others to derive the dynamics equations. We found *CasADi*'s graph optimization to greatly improve evaluation times of the algebraically involved expressions resulting from the kinematic loops. To specify the optimization problem we used *CasADi*'s *Opti* framework, which allows to export the problem programming language independent. We remark that enabling the compilation option of *CasADi* reduced optimization times up to three times—at the cost of a one-time compilation duration in the order of minutes.

For modeling the 2-D terrain, we chose the functions $c_i(x)$ to lie in $\mathcal{C}^5$ in order for the Jacobians and Hessians of the optimization problem to be smooth, which is desirable when employing gradient-based solvers.

2) Solver: To solve the proposed optimization problem, we evaluated several state-of-the-art NLP solvers.[13] We note that a qualitative or even quantitative comparison, including the tuning of each solver's settings and the use of comparable linear system solvers, is outside the scope of this work. The listing is in the order of decreasing convergence speed and success as follows.

- *IPOPT [79]:* Convergence was achieved for all test cases. As linear system solver we used the *HSL's MA97* routine, which compared to the default *MA27* routine reduced solver times by up to a factor of two. With default *IPOPT* solver settings and a reasonable discretization of roughly 20 collocation intervals per phase, trajectory generation for a terrain consisting of a single step (which results in three phases) took 0.3 s only. For more complex terrain with 4–8 phases, solver times ranged between 2–4 s. Approximately 15 s were required for a staircase with ten steps (>20 phases). By tweaking the solver settings we could further half solution times[14]—especially the option of using a Hessian approximation turned out to be beneficial, as Hessian evaluations proved to be the most expensive part of the optimization.

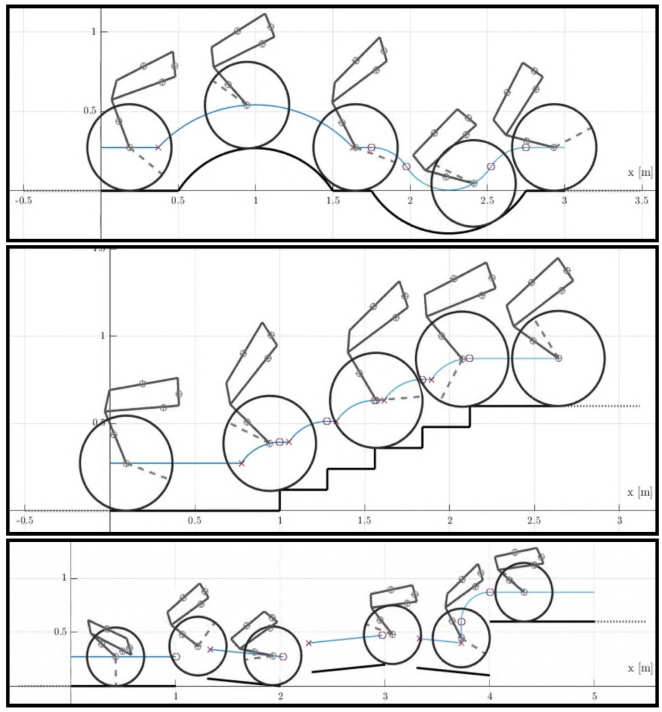


Fig. 7. Three selected trajectories over different terrains. *Top:* Terrain featuring a half-pipe and an inverted half-pipe. We note the impacts when transitioning from a straight to a curved surface and vice versa. *Middle:* An entire stairway. We note that momentum accumulated while approaching the stair is used for overcoming the first three steps, whereas the remaining ones are overcome by a limit cycle-like motion. *Bottom:* A jumping course. We note the use of the wheel's rotational inertia during flight phases to reorient the system.

- *SNOPT [80]:* Convergence was achieved for all test cases. The solver times were faster than *IPOPT*'s for smaller problems, but significantly slower for larger ones.
- *WORHP [81]:* Convergence was achieved for all test cases. The solver times were in general slower (by roughly 50%) than *IPOPT*'s with default settings.
- *Structure-Exploiting SQP ("Scpgen") [78], [82]:* Convergence was achieved only for test cases with short horizons.[15] The solver times were in general slower than *WORHP*'s.
- *Bare bone SQP ("Sqpmethod") [78]:* Convergence was achieved for none of the test cases.

Overall, we were satisfied most with *IPOPT* and therefore made it our principal solver.

3) Initial Guessing Procedure: For initializing the primal variables we resorted to the simple procedure of linearly interpolating between the initial and terminal positional states of the optimization, whereby the durations of the phases were chosen proportional to their arc lengths. For each phase we then set the initial guess of the arc velocity equal to the phase's arc length

[12] [Online]. Available: https://youtu.be/vWkOERAH_zI.

[13] For the evaluations we used a *NUC8i7HVK* with a quad-core *i7-8809 G* processor, with up to eight threads and a processor frequency of 3.10–4.20 GHz.

[14] The trajectory quality slightly deteriorated, however. Namely, the curves during the phases were not as smooth in that case.

[15] We note that identifying the cause of the convergence failures would require a thorough analysis of the respective solvers, which is outside the scope of this work. They are probably not *globally convergent*, that is, given an arbitrary starting point they do not necessarily converge to a local minimum. In the case of sequential quadratic programming (SQP) methods it can for instance occur that, although the optimization problem is feasible, its linearized version becomes infeasible [83].

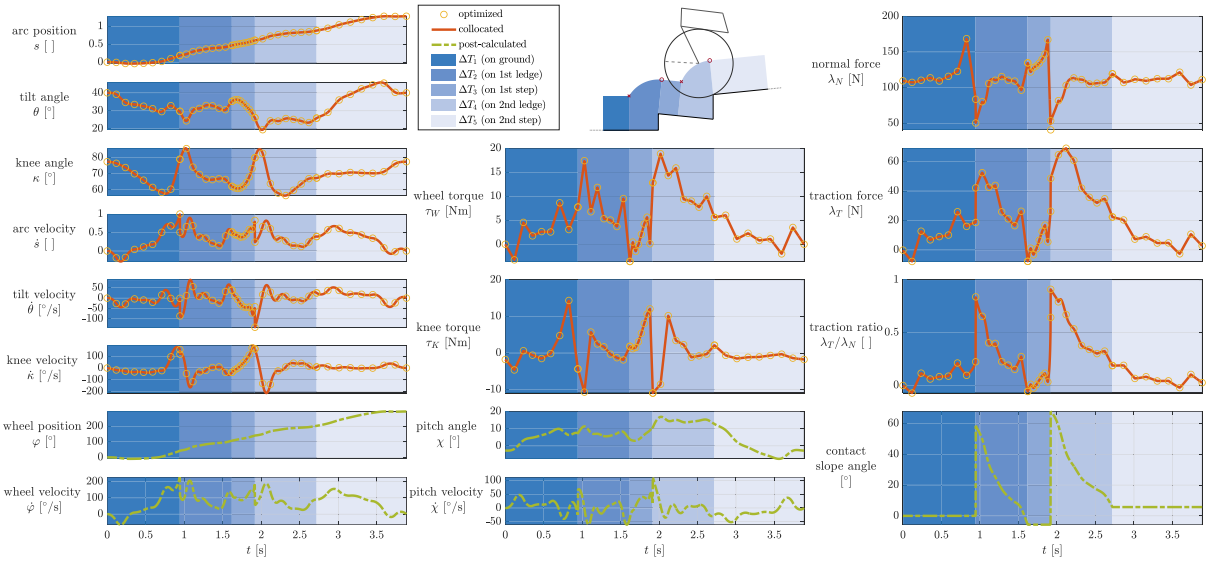


Fig. 8. Complete set of optimization variables defining a trajectory for overcoming two sloped steps with heights of 12 and 17 cm, respectively. Furthermore, the resulting wheel angle, base pitch angle, traction ratio, and contact slope angle are provided, since they are useful for interpretation of the trajectory from a physical point of view. The contact slope angle denotes the angle between the horizontal and the tangent vector at the point of the wheel currently in contact with the ground. We note that impulses in the torques and contact forces are not shown, since their calculation can be circumvented by the impact projection (12) and they thus do not explicitly occur in our modeling. Rather, we connect the right- and left-hand limits by a straight line.

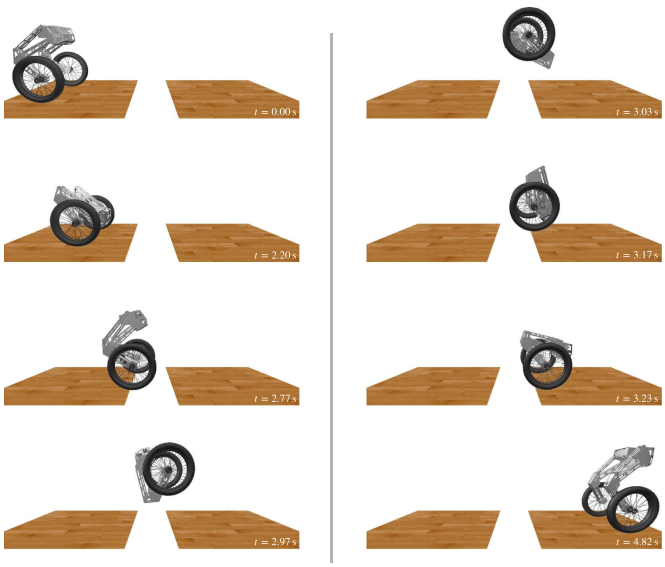


Fig. 9. Visualization of a backflip trajectory. We remark that the trajectory leverages the rotational inertia of the wheels during flight, i.e., they are used alike reaction wheels. Namely, they are first accelerated to spin quickly such as to assist the rotation of the base, before decelerating such as to attain a suitable velocity for landing.

divided by the initial guess of the corresponding phase duration. Furthermore, we initialized the knee torque to its steady state value.

B. Examples of Trajectory Solutions

To evaluate the versatility of the TO scheme we generated trajectories over more than 50 different terrains, including steps, stairs, ramps, trenches, half-pipes, bumps, and combinations thereof. Failure to return a solution only seemed to occur when the terrain in question would not permit a physical solution, e.g., due to actuator limitations in the case of jumps higher than 1.2 m. Three exemplary trajectories over complex terrain are presented in Fig. 7.

We show in detail the solution for overcoming two sloped steps in Fig. 8. The trajectory comprises a total of five contact phases with two impactive and two smooth transitions. The state, input, and contact force constraints are satisfied at all times. The traction ratio achieves its maximum right when hitting the ledges of the steps. We note the impacts seen for the arc parameter, tilt, knee, and wheel velocities at those time instances.

As mentioned in Section III-E, it is possible to let the robot perform a backflip during a jump over a terrain gap simply by increasing the desired terminal value for the pitch angle by 2π, see Fig. 9.

C. Robustness Evaluation

To gain insight into the robustness of our trajectory optimizer when used in conjunction with the tracking controller proposed in Section IV-B, we conducted a number of tests in simulation.[16] Specifically, we evaluated the performance against various types of adverse effects: impactive disturbances, process noise, actuation noise, modeling errors, measurement noise, measurement delays, actuation delays, and terrain mismatch. Of these, actuation delay turned out to have the most detrimental effect. However, similar to most of the other adverse effects, we did not observe it to be of concern on the real system. For space reasons we therefore do not present the corresponding

[16] A custom, phase-based 2-D simulator was employed, which detects contact switches and relies on the modeling outlined in Sections II-B, II-C, and II-D. Also, the tracking controller was adapted to the 2-D case.

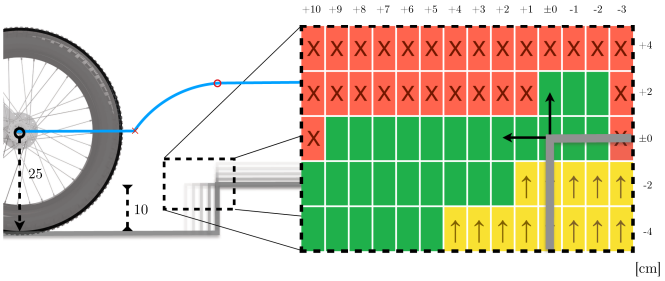


Fig. 10. Trajectory for overcoming a 10 cm step is evaluated for its robustness against terrain variation. Applying the same trajectory for varying step heights and horizontal positions yielded the robustness map shown on the right. Entries are colored green when climbing the step was successful, red when it failed (marked with a cross), and yellow when undesired lift-off occurred (marked with an arrow).

evaluations and instead focus on the practically most significant case of terrain mismatch. Understanding its influence is not only relevant because the geometry of the terrain and the relative robot location are difficult to perceive accurately, but also because we found it beneficial to overestimate the height of steps, as described in the next section.

To this end, we modified the height and position of a 10 cm step after the trajectory had been generated. Fig. 10 shows the robustness map obtained from executing the trajectory over the varied terrains. The results suggest that overestimating the step height does not represent much of an issue, though undesired liftoff may take place. This occurs in particular when the distance to the step is simultaneously underestimated, since this causes premature retraction of the legs. These findings match experimental results on the real system qualitatively very well and explain the robustness benefit from overestimating the step height in the TO.

D. Real-World Experimental Validation

We conclude this section by presenting several real-world experiments on the two latest *Ascento* prototypes.

1) Experimental Setup: Both prototypes are similar in design and use the same motors. The newer one has a more rigid mechanical design, at the cost of a weight increase of 18%. In addition, it features a completely redesigned spring mechanism, which omits torsional springs.[17] Thanks to the model-based approach, switching between prototypes required little more than changing the inertial parameters and performing the identification procedures described in Section IV-C.

The tracking controller and the state estimator are implemented in *ROS* [84] with C++ using *Eigen* [85] and the trajectories are made available through the C++ *CasADi* interface: Either the trajectories are computed in a separate thread, or they are precomputed and loaded for a specific situation. This allows the tracking controller to maintain a constant frequency of 400 Hz. To solve the discrete time algebraic Ricatti equation used in the tracking controller, we use the robust iterative solver from [86] and initialize it with the Riccati matrix from the previous iteration. This leads to solution times of less than 0.25 ms, which is far from being critical for the required 400 Hz of the control loop.

2) Trajectory Tracking Experiments: The following is a presentation of selected trajectory tracking experiments, most of which are shown in Fig. 11. The execution of trajectories over more complex terrain appears to be limited mostly by the proprioceptive state estimation, which impedes accurate estimation of the robot's position relative to the terrain over time.

- *Single Step:* Up to a height of 20 cm—which is close to the wheel radius of 25 cm—steps could be reliably overcome. The case of a 10 cm and a 20 cm step are shown in Fig. 11. We also tested how the initial distance (which was accounted for in the trajectory optimization) would influence reliability. Not very surprisingly, given the proprioceptive setup, it turned out to be most reliable to execute trajectories which start with the wheels touching the step (but where driving backward is permitted for accelerating, cf., Footnote 7). This ensures that the robot's position relative to the terrain is initially precisely known. Furthermore, steps were the only terrain type for which we adjusted parameters for better robustness. Namely, overestimating the step height by 20–50% (higher values caused undesired liftoff) in the TO resulted in a near 100% success rate.[18] We attribute this need for compensation to energy dissipation caused by the deformation of the tires.
- *Stand up: Ascento* is able to get into a resting position by sitting on its bent knees to save energy when idling. Getting up again, however, is nontrivial. Our TO enabled us to find a trajectory to return from this position to a balancing state.[19] It worked successfully at the first attempt and could be repeatedly executed without a single failure.
- *Jumps:* To validate the ability of tracking trajectories containing flight phases, we executed jumps on spot with a desired ground clearance of 5 cm. Two cases were tested: setting the torque limit to 40 Nm and to 30 Nm. Both could be executed repeatedly without failure, the latter showing much more use of momentum by leveraging the system's leg dynamics.[20]
- *Two Steps:* Fig. 12 shows the tracking of the wheel, knee, and pitch angles when executing a trajectory for climbing up two 10 cm steps in sequence. We note the satisfaction of the traction limits as indicated by the absence of wheel slip.

3) Staircase Climbing: With the current proprioceptive state estimation, tracking of a single trajectory over a full staircase can fail due to inaccurate localization w.r.t. the terrain over time. Nevertheless, traversal of stairs with arbitrary numbers

[17] The tracking performance on the newer prototype was clearly superior. As outlined in Section IV-C, we attribute this to reduced friction, among others.

[18] However, even when the robot would fail to climb the step, it would never lose balance, but simply roll down the ledge again. Continuous testing sessions comprising a total of 96 trajectory experiments did not result in a single fall.

[19] We modeled this problem by implicitly assuming that contact between the knee and the ground is broken at the first time step, i.e., that the trajectory starts in a phase where only the wheel is in contact with the ground.

[20] We also attempted jumping over a gap. While this did work, it unfortunately caused mechanical damage to one of the motors due to weak internal design. We therefore suspended further testing of jumps.

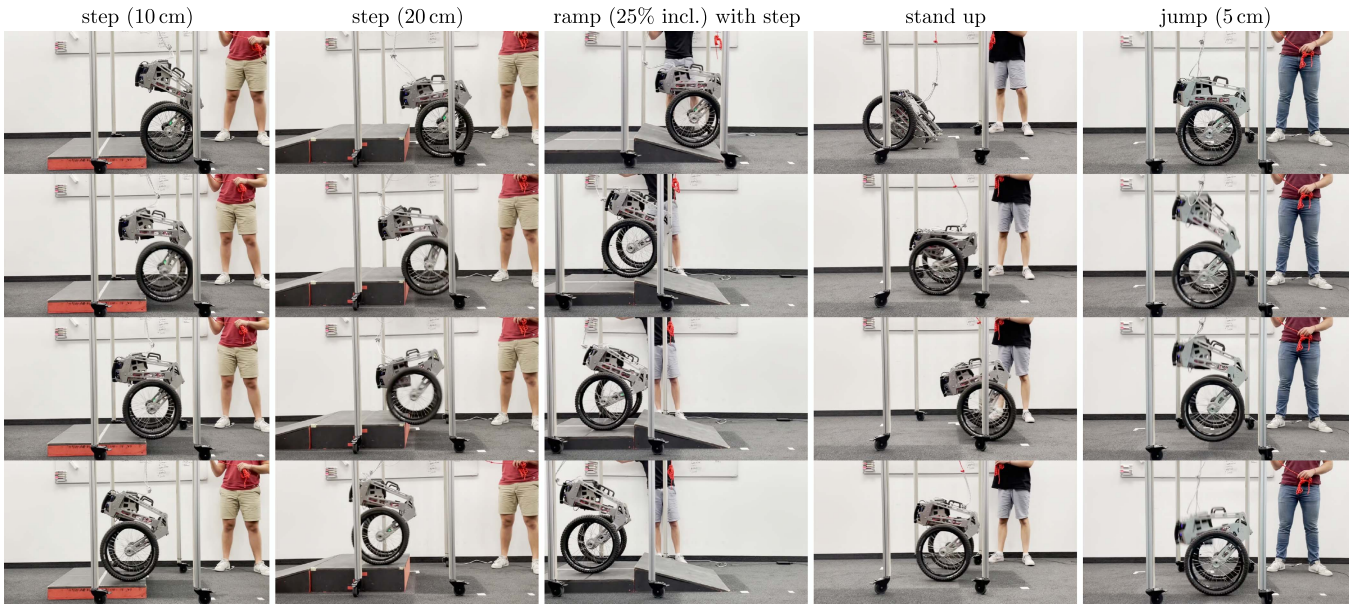


Fig. 11. Snapshot sequences (top to bottom) of various trajectories tracked on the real prototype.

of steps can be achieved by successive execution of single step trajectories which are triggered when the next ledge is hit. To ensure that contact to the ledge is actually established and maintained until the tracking of the next trajectory starts, we superimposed a small forward velocity of 0.3 m/s. This has the additional advantages of causing the robot to self-align and of mitigating the risk of rolling back onto the previous step. With this setup we were able to climb stairs, such as the one shown in Fig. 1. We note that given the step width of 29 cm and the wheel radius of 25 cm, only little room was available for the rebalancing and acceleration maneuvers.

4) Tracking Control Without TO: The tracking controller on its own proved to be exceedingly capable in stabilizing the robot and preventing it from losing balance. The robustness is nicely demonstrated by driving over highly unstructured terrain, such as forest ground, where the control references are provided by the user via remote-control instead of the TO—see the accompanying video. Moreover, the tracking controller alone is sufficient for reliably driving down staircases (but not up), which is also shown in the video. Overall, we never experienced the case that failure to track a trajectory would result in the robot falling over.

In combination with the robustness evaluation of the preceding section, these experiments permit some indications as to the importance of accurate terrain mapping and localization. First, the robustness evaluation suggests that neither the dimensions of the terrain nor the relative location of the robot must be known perfectly. In particular, by feeding the TO with appropriately shifted dimensions, convenient safety margins may be obtained. Second, the last experiment suggests that a certain degree of smoothing and lack of resolution in the terrain mapping can be compensated for. Third, the effectiveness of the tracking controller in preventing loss of balance reduces the necessity of accurate perception even further because after a failed attempt to traverse an obstacle the robot may simply try again. The subsequent outlook complements these considerations by proposing extensions that may contribute to successful autonomous deployment in real-world applications.

VI. Conclusion

In this work, we have proposed a trajectory optimizer for locomotion of wheeled balancing robots over a large class of challenging terrains. The presented scheme is able to handle contact switches and impacts and generates nonsmooth trajectories that leverage the full 2-D rigid-body dynamics while respecting explicit traction and input constraints. We applied our approach to the *Ascento* robot and, as part of this worked example, provided a new closed-form solution of its dynamics equations.

Our simulation results do not only confirm the versatility of our method, but also indicate that the optimized trajectories are suitable for application under the presence of moderate uncertainties and model-mismatches. Indeed, experiments with the latest *Ascento* prototype, which features purely proprioceptive state-estimation, demonstrate that trajectories for jumping, standing up, and overcoming steps, among others, can successfully be executed. Most importantly, by sequentially triggering the execution of single step trajectories, climbing of standard stairways was enabled.

As an outlook, we list a selection of extensions that could constitute promising directions of research.

1) *Time-Varying Tracking LQR*: As some preliminary tests suggested, [21] a feedback law that varies over the course of

[21] We computed a time-varying LQR for some cases, which revealed an interesting property: When driving up a step, there is a sign reversal on the wheel angle gain just before hitting the ledge. This indicates that at a certain

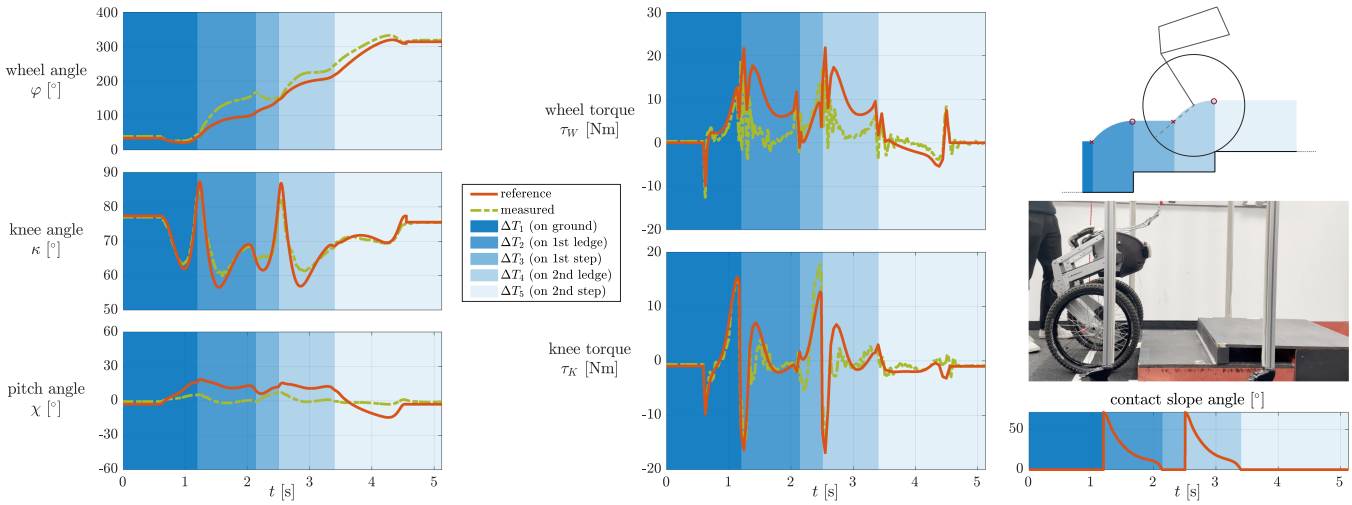


Fig. 12. Tracking (*green*) of a trajectory (*orange*) for climbing two 10 cm steps in a row. In particular on the ledge, we note a correction of the wheel torque by the tracking controller to stabilize the pitch angle. The deviation of the latter is likely required to compensate for model errors, which we mainly attribute to COM position errors, rolling friction, and contact point shifts due to deformation of the soft tires on the ledge.

the trajectory could further improve robustness and tracking performance. Unfortunately, such gain scheduling methods are notorious for demanding very high actuator bandwidths. The frequency-aware cost shaping proposed by [35], however, could alleviate this problem.

2) *Replanning:* Applying the TO in a receding horizon fashion could considerably improve robustness and traversal of long, complex terrain segments. Warm-starting could then be expected to significantly lower average computation times, since primal and dual variables should essentially be shifted in time. This could be combined with the use of an offline motion library, as proposed in [7]. In addition, a much faster replanning loop—potentially employing a shorter horizon and a simplified model—could be introduced, targeting a frequency of 20–30 Hz (for these numbers see [31], [35], [87]). Contact schedules could be taken over from the high fidelity, long horizon TO, similar to [7], [47].

3) *Extension to 3-D:* There are several ways in which an extension to handle 3-D terrain could be attempted. For the example of *Ascento*, a straightforward approach—probably already covering a majority of use cases—would be to assume the roll and yaw motion as a function of the arc parameter to be given, for instance by a high-level planner. Essentially, the modeling would then remain planar, but take place on a curved 2-D surface. Regarding the possibility that the individual wheels may encounter different terrains, we note that based on our experience this should only present an obstacle when the difference is substantial. The effects of small differences are well compensated by the tracking controller, while large differences could probably often be circumvented by a dedicated planner.

4) *Event-Based State Estimation:* To overcome the limitations imposed by the current proprioceptive setup without requiring a more complex sensory system, the information acquired from the occurrence of impacts could be leveraged. Illustrated by the example of hitting the ledge of a step, these could serve as priors for the state estimation to locate the robot relative to the terrain. This would likely enable the execution of much longer trajectories and bring the system a good step closer to autonomous operation.

point it is more optimal to accelerate forwards and handle the rebalancing after surmounting the step.

APPENDIX

A. Kinematically Consistent Collocation Scheme

In the following, we provide the derivation of the extended Hermite–Simpson collocation scheme proposed in Section III-C. Our starting point is a vector of third order polynomials $\boldsymbol{p}_{\dot{q}}$, which serves to interpolate the generalized velocities $\dot{\boldsymbol{q}}$ over an interval of duration Δt. Given its representation in Hermite form, that is, by the coefficients $\dot{\boldsymbol{q}}_A = \boldsymbol{p}_{\dot{q}}(0)$, $\dot{\boldsymbol{q}}_B = \boldsymbol{p}_{\dot{q}}(\Delta t)$, $\ddot{\boldsymbol{q}}_A = \dot{\boldsymbol{p}}_{\dot{q}}(0)$, $\ddot{\boldsymbol{q}}_B = \dot{\boldsymbol{p}}_{\dot{q}}(\Delta t)$, we have that

$$\begin{aligned}\boldsymbol{p}_{\dot{q}}(t) = &\left(\frac{1}{\Delta t^2}\left(\ddot{\boldsymbol{q}}_A + \ddot{\boldsymbol{q}}_B\right) - \frac{2}{\Delta t^3}\left(\dot{\boldsymbol{q}}_B - \dot{\boldsymbol{q}}_A\right)\right) t^3 \\ &+ \left(\frac{3}{\Delta t^2}\left(\dot{\boldsymbol{q}}_B - \ddot{\boldsymbol{q}}_A\right) - \frac{2}{\Delta t}\left(\dot{\boldsymbol{q}}_A + \dot{\boldsymbol{q}}_B\right)\right) t^2 \\ &+ \ddot{\boldsymbol{q}}_A\, t + \dot{\boldsymbol{q}}_A.\end{aligned}$$

Evaluation at the midpoint $\frac{\Delta t}{2}$ gives (15) and (16):

$$\dot{\boldsymbol{q}}_C = \boldsymbol{p}_{\dot{q}}\left(\frac{\Delta t}{2}\right) = \frac{1}{2}\left(\dot{\boldsymbol{q}}_A + \dot{\boldsymbol{q}}_B\right) + \frac{\Delta t}{8}\left(\ddot{\boldsymbol{q}}_A - \ddot{\boldsymbol{q}}_B\right)$$

$$\ddot{\boldsymbol{q}}_C = \dot{\boldsymbol{p}}_{\dot{q}}\left(\frac{\Delta t}{2}\right) = \frac{-3}{2\,\Delta t}\left(\dot{\boldsymbol{q}}_A - \dot{\boldsymbol{q}}_B\right) - \frac{1}{4}\left(\ddot{\boldsymbol{q}}_A + \ddot{\boldsymbol{q}}_B\right).$$

We note that these are the standard midpoint formulas of the cubic Hermite–Simpson collocation scheme applied to $\dot{\boldsymbol{q}}$.

To obtain a polynomial $\boldsymbol{p}_q$ interpolating the generalized coordinates $\boldsymbol{q}$ over the interval $[0, \Delta t]$ which is kinematically consistent with the polynomial approximation of $\dot{\boldsymbol{q}}$, we define $\boldsymbol{p}_q$ as the integral of $\boldsymbol{p}_{\dot{q}}$, that is:

$$\begin{aligned}\boldsymbol{p}_q(t) &= \int_0^t \boldsymbol{p}_{\dot{q}}(\tau)\, d\tau \\ &= \left(\frac{1}{4\Delta t^2} (\ddot{\boldsymbol{q}}_A + \ddot{\boldsymbol{q}}_B) - \frac{1}{2\Delta t^3} (\dot{\boldsymbol{q}}_B - \dot{\boldsymbol{q}}_A) \right) t^4 \\ &\quad + \left(\frac{1}{\Delta t^2} (\dot{\boldsymbol{q}}_B - \ddot{\boldsymbol{q}}_A) - \frac{2}{3\Delta t} (\dot{\boldsymbol{q}}_A + \dot{\boldsymbol{q}}_B) \right) t^3 \\ &\quad + \frac{1}{2}\ddot{\boldsymbol{q}}_A\, t^2 + \dot{\boldsymbol{q}}_A\, t + \boldsymbol{q}_A. \end{aligned} \tag{24}$$

Evaluation at the midpoint $\frac{\Delta t}{2}$ gives (14):

$$\begin{aligned}\boldsymbol{q}_C = \boldsymbol{p}_q\left(\frac{\Delta t}{2}\right) &= \frac{\Delta t}{32} (13\, \dot{\boldsymbol{q}}_A + 3\, \dot{\boldsymbol{q}}_B) \\ &\quad + \frac{\Delta t^2}{192} (11\, \ddot{\boldsymbol{q}}_A - 5\, \ddot{\boldsymbol{q}}_B) + \boldsymbol{q}_A. \end{aligned} \tag{25}$$

As in the standard cubic Hermite–Simpson scheme, the dynamics are required to be satisfied at the midpoint, which gives us the (differential) collocation constraint (17):

$$\ddot{\boldsymbol{q}}_C = \boldsymbol{f}_{\ddot{\boldsymbol{q}}}(\boldsymbol{q}_C, \dot{\boldsymbol{q}}_C, \boldsymbol{\tau}_C)$$

where $\boldsymbol{\tau}_C = \frac{1}{2}(\boldsymbol{\tau}_A + \boldsymbol{\tau}_B)$, with $\boldsymbol{\tau}_A = \boldsymbol{\tau}(0)$ and $\boldsymbol{\tau}_B = \boldsymbol{\tau}(\Delta t)$.

As the attentive reader might have noticed, there is no dependency on $\boldsymbol{q}_B$ in (25), which must be established by an additional constraint.[22] Requiring (24) to match $\boldsymbol{q}_B$ at Δt gives us the (integral) collocation constraint (18):

$$\boldsymbol{q}_B = \boldsymbol{p}_q(\Delta t) = \frac{\Delta t}{2} (\dot{\boldsymbol{q}}_A + \dot{\boldsymbol{q}}_B) + \frac{\Delta t^2}{12} (\ddot{\boldsymbol{q}}_A - \ddot{\boldsymbol{q}}_B) + \boldsymbol{q}_A.$$

This completes the derivation.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Ciro Salzmann, Lionel Gulich, Alessandro Morra, and Dominik Mannhart for their support in real-world testing and recording of the accompanying video, as well as all supporters of the Ascento project.

REFERENCES

[1] M. Tranzatto et al., "CERBERUS in the DARPA subterranean challenge," *Sci. Robot.*, vol. 7, no. 66, 2022, Art. no. eabp9742.
[2] R. Siegwart and I. R. Nourbakhsh, *Autonomous Mobile Robots*. Cambridge, MA, USA: MIT Press, 2004.
[3] C. Zheng and K. Lee, "WheeLeR: Wheel-leg reconfigurable mechanism with passive gears for mobile robot applications," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 9292–9298.
[4] I. Kim, W. Jeon, and H. Yang, "Design of a transformable mobile robot for enhancing mobility," *Int. J. Adv. Robot. Syst.*, vol. 14, no. 1, 2017, Art. no. 1729881416687135.

[22] Applying the cubic Hermite–Simpson scheme in the standard way to our second-order dynamical system, we would have interpolated the generalized coordinates by a vector of cubic polynomials $\tilde{\boldsymbol{p}}_q$ and imposed at the midpoint the (differential) constraint $\dot{\tilde{\boldsymbol{p}}}_q(\frac{\Delta t}{2}) = \boldsymbol{p}_{\dot{q}}(\frac{\Delta t}{2})$, which in our case is replaced by the following one.

[5] M. H. Hoepflinger et al., "ParcelBot: A tracked parcel transporter with high obstacle negotiation capabilities," in *Adaptive Mobile Robotics*. Singapore: World Scientific, 2012, pp. 831–838.
[6] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, "Whole-body MPC and online gait sequence generation for wheeled-legged robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8388–8395.
[7] M. Bjelonic et al., "Offline motion libraries and online MPC for advanced mobility skills," *Int. J. Robot. Res.*, vol. 41, no. 9/10, pp. 903–924, 2022, doi: 10.1177/02783649221102473.
[8] M. Bjelonic, V. Klemm, J. Lee, and M. Hutter, "A survey of wheeled-legged robots," in *Robotics in Natural Settings*, J. M. Cascalho, M. O. Tokhi, M. F. Silva, A. Mendes, K. Goher, and M. Funk, Eds. Cham, Switzerland: Springer, 2023, pp. 83–94.
[9] S. Wang et al., "Balance control of a novel wheel-legged robot: Design and experiments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 6782–6788.
[10] C. Zhang, T. Liu, S. Song, and M. Q.-H. Meng, "System design and balance control of a bipedal leg-wheeled robot," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2019, pp. 1869–1874.
[11] V. Klemm et al., "Ascento: A two-wheeled jumping robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 7515–7521.
[12] H. Chen, B. Wang, Z. Hong, C. Shen, P. M. Wensing, and W. Zhang, "Underactuated motion planning and control for jumping with wheeled-bipedal robots," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 747–754, Apr. 2021.
[13] F. Raza, A. Chemori, and M. Hayashibe, "A new augmented L1 adaptive control for wheel-legged robots: Design and experiments," in *Proc. Amer. Control Conf.*, 2022, pp. 22–27.
[14] Boston Dynamics, "Introducing handle," Youtube, 2017. [Online]. Available: https://youtu.be/-7xvqQeoA8c
[15] Boston Dynamics, "Handle robot reimagined for logistics," Youtube, 2019. [Online]. Available: https://youtu.be/5iV_hB08Uns
[16] IEEE Spectrum, "Tencent's new wheeled robot flicks its tail to do backflips," 2021. [Online]. Available: https://spectrum.ieee.org/tencents-new-wheeled-robot-flicks-its-tail-to-do-backflips
[17] V. Klemm et al., "LQR-assisted whole-body control of a wheeled bipedal robot with kinematic loops," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3745–3752, Apr. 2020.
[18] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Philadelphia, PA, USA: SIAM, 2010.
[19] J. Hauser and A. Saccon, "Motorcycle modeling for high-performance maneuvering," *IEEE Control Syst. Mag.*, vol. 26, no. 5, pp. 89–105, Oct. 2006.
[20] Z. Li, C. Yang, and L. Fan, *Advanced Control of Wheeled Inverted Pendulum Systems*. London, U.K.: Springer, 2012.
[21] C. Glocker, *Set-Valued Force Laws: Dynamics of Non-Smooth Systems* (Lecture Notes in Applied and Computational Mechanics Series). Berlin, Germany: Springer2001.
[22] B. Brogliato, *Nonsmooth Mechanics: Models, Dynamics and Control* (Communications and Control Engineering Series), 3rd ed. Cham, Switzerland: Springer, 2016.
[23] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Algorithmic Foundations of Robotics X*. Berlin, Germany: Springer, 2013, pp. 527–542.
[24] A. Patel, S. L. Shield, S. Kazi, A. M. Johnson, and L. T. Biegler, "Contact-implicit trajectory optimization using orthogonal collocation," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2242–2249, Apr. 2019.
[25] S. Shield, A. M. Johnson, and A. Patel, "Contact-implicit direct collocation with a discontinuous velocity state," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5779–5786, Apr. 2022.
[26] A. Werner, W. Turlej, and C. Ott, "Generation of locomotion trajectories for series elastic and viscoelastic bipedal robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5853–5860.
[27] K. Yunt and C. Glocker, "Trajectory optimization of mechanical hybrid systems using SUMT," in *Proc. IEEE 9th Int. Workshop Adv. Motion Control*, 2006, pp. 665–671.
[28] T. A. Howell, K. Tracy, S. Le Cleac'h, and Z. Manchester, "CALIPSO: A differentiable solver for tajectory optimization with conic and complementarity constraints," in *Robotics Research*, A. Billard, T. Asfour, and O. Khatib, Eds. Cham, Switzerland: Springer, 2023, pp. 504–521.
[29] Z. Manchester, N. Doshi, R. J. Wood, and S. Kuindersma, "Contact-implicit trajectory optimization using variational integrators," *Int. J. Robot. Res.*, vol. 38, no. 12/13, pp. 1463–1476, May 2019.
[30] S. L. Cleac'h, T. Howell, M. Schwager, and Z. Manchester, "Fast contact-implicit model-predictive control," 2021, *arXiv:2107.05616*.

[31] M. Neunert et al., "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018.

[32] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3402–3421, Oct. 2023.

[33] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "TAMOLS: Terrain-aware motion optimization for legged systems," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3395–3413, Dec. 2022.

[34] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2019, pp. 4730–4737.

[35] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "Frequency-aware model predictive control," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1517–1524, Apr. 2019.

[36] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[37] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT Press, 1986.

[38] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2261–2268, Jul. 2018.

[39] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, "Fast trajectory optimization for legged robots using vertex-based ZMP constraints," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2201–2208, Oct. 2017.

[40] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effectorparameterization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1560–1567, Jul. 2018.

[41] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, no. 4, Jul. 2012, Art. no. 43.

[42] M. Egerstedt, Y. Wardi, and H. Axelsson, "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. Autom. Control*, vol. 51, no. 1, pp. 110–115, Jan. 2006.

[43] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "SkaterBots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018, Art. no. 160.

[44] M. Geilinger, S. Winberg, and S. Coros, "A computational framework for designing skilled legged-wheeled robots," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3674–3681, Apr. 2020.

[45] M. Bjelonic et al., "Keep rollin'—Whole-body motion control and planning for wheeled quadrupedal robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2116–2123, Apr. 2019.

[46] Y. de Viragh, M. Bjelonic, C. D. Bellicoso, F. Jenelten, and M. Hutter, "Trajectory optimization for wheeled-legged quadrupedal robots using linearized ZMP constraints," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1633–1640, Apr. 2019.

[47] M. Bjelonic, P. K. Sankar, C. D. Bellicoso, H. Vallery, and M. Hutter, "Rolling in the deep–hybrid locomotion for wheeled-legged robots using online trajectory optimization," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3626–3633, Apr. 2020.

[48] V. S. Medeiros, E. Jelavic, M. Bjelonic, R. Siegwart, M. A. Meggiolaro, and M. Hutter, "Trajectory optimization for wheeled-legged quadrupedal robots driving in challenging terrain," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4172–4179, Jul. 2020.

[49] J. Li, J. Ma, and Q. Nguyen, "Balancing control and pose optimization for wheel-legged robots navigating uneven terrains," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 8835–8841.

[50] P. Jarrault, C. Grand, and P. Bidaud, "Robust obstacle crossing of a wheel-legged mobile robot using minimax force distribution and self-reconfiguration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2011, pp. 2753–2758.

[51] K. Turker, I. Sharf, and M. Trentini, "Step negotiation with wheel traction: A strategy for a wheel-legged robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 1168–1174.

[52] F. Cordes, A. Babu, and F. Kirchner, "Static force distribution and orientation control for a rover with an actively articulated suspension system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5219–5224.

[53] G. Zambella et al., "Dynamic whole-body control of unstable wheeled humanoid robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3489–3496, Oct. 2019.

[54] M. Zafar, S. Hutchinson, and E. A. Theodorou, "Hierarchical optimization for whole-body control of wheeled inverted pendulum humanoids," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 7535–7542.

[55] F. Raza, D. Owaki, and M. Hayashibe, "Modeling and control of a hybrid wheeled legged robot: Disturbance analysis," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, 2020, pp. 466–473.

[56] S. Xin and S. Vijayakumar, "Online dynamic motion planning and control for wheeled biped robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2020, pp. 3892–3899.

[57] T. Dinev, S. Xin, W. Merkt, V. Ivan, and S. Vijayakumar, "Modeling and control of a hybrid wheeled jumping robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2020, pp. 2563–2570.

[58] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, 2020, Art. no. eabc5986.

[59] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, 2022, Art. no. eabk2822.

[60] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Learning humanoid locomotion with transformers," 2023, *arXiv:2303.03381*.

[61] J. Lee, M. Bjelonic, and M. Hutter, "Control of wheeled-legged quadrupeds using deep reinforcement learning," in *Robotics in Natural Settings*, J. M. Cascalho, M. O. Tokhi, M. F. Silva, A. Mendes, K. Goher, and M. Funk, Eds. Cham, Switzerland: Springer, 2023, pp. 119–127.

[62] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter, "Advanced skills through multiple adversarial motion priors in reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 5120–5126.

[63] L. Cui et al., "Learning-based balance control of wheel-legged robots," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7667–7674, Oct. 2021.

[64] W. Zhu, F. Raza, and M. Hayashibe, "Reinforcement learning based hierarchical control for path tracking of a wheeled bipedal robot with Sim-to-Real framework," in *Proc. IEEE/SICE Int. Symp. Syst. Integration*, 2022, pp. 40–46.

[65] J. Zhang et al., "Adaptive optimal output regulation for wheel-legged robot Ollie: A data-driven approach," *Front. Neurorobot.*, vol. 16, 2022, Art. no. 1102259.

[66] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," 2019, *arXiv:1901.07517*.

[67] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2022, pp. 2497–2503.

[68] D. Hoeller, N. Rudin, C. Choy, A. Anandkumar, and M. Hutter, "Neural scene representation for locomotion on structured terrain," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 8667–8674, Oct. 2022.

[69] T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, and M. Hutter, "Elevation mapping for locomotion and navigation using GPU," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 2273–2280.

[70] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control* (Advanced Textbooks in Control and Signal Processing Series). London, U.K.: Springer2010.

[71] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Cham, Switzerland: Springer, 2016.

[72] R. Tedrake, "Underactuated robotics: Learning, planning, and control for efficient and agile machines," Course Notes for MIT 6.832, Massachusetts Institute of Technology, Cambridge, MA, USA, Working draft edition, 2009.

[73] B. Katz, J. Di Carlo, and S. Kim, "Mini Cheetah: A platform for pushing the limits of dynamic quadruped control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 6295–6301.

[74] V. Klemm, D. Mannhart, and R. Siegwart, "Design optimization of a four-bar leg linkage for a legged-wheeled balancing robot," in *Robotics in Natural Settings*, J. M. Cascalho, M. O. Tokhi, M. F. Silva, A. Mendes, K. Goher, and M. Funk, Eds. Cham, Switzerland: Springer, 2023, pp. 128–139.

[75] N. J. Kong, G. Council, and A. M. Johnson, "iLQR for piecewise-smooth hybrid dynamical systems," in *Proc. 54th IEEE Conf. Decis. Control*, 2021, pp. 5374–5381.

[76] M. Rijnen, A. Saccon, and H. Nijmeijer, "Reference spreading: Tracking performance for impact trajectories of a 1DoF setup," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 3, pp. 1124–1131, May 2019.

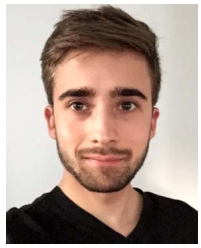
[77] W. Shaowei and W. Shanming, "Velocity and acceleration computations by single-dimensional Kalman filter with adaptive noise variance," *Przeglad Elektrotechniczny*, vol. 88, no. 2, pp. 283–287, 2012.
[78] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADI: A software framework for nonlinear optimization and optimal control," *Math. Program. Computation*, vol. 11, no. 1, pp. 1–36, 2019.
[79] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
[80] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, no. 1, pp. 99–131, 2005.
[81] T. Nikolayzik, C. Büskens, and M. Gerdts, "Nonlinear large-scale optimization with WORHP," in *Proc. 13th AIAA/ISSMO Multidisciplinary Anal. Optim. Conf.*, 2010, Paper 9136.
[82] J. Andersson and M. Diehl, "A general-purpose software framework for dynamic optimization," Ph.D. dissertation, KU Leuven, Leuven, Belgium, 2013.
[83] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming* (International Series in Operations Research & Management Science), 5th ed. Cham, Switzerland: Springer, 2021.
[84] M. Quigley et al., "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, 2009, vol. 3, no. 3.2.
[85] G. Guennebaud et al., "Eigen v3," 2010. [Online]. Available: http://eigen.tuxfamily.org
[86] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli, "The control toolbox - An open-source C library for robotics, optimal and model predictive control," in *Proc. IEEE Int. Conf. Simul., Model., Program. Auton. Robots*, 2018, pp. 123–129.
[87] G. Bledt and S. Kim, "Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 6316–6323.

Victor Klemm (Graduate Student Member, IEEE) received the B.Sc. degree in mechanical engineering and the M.Sc. degree in robotics, systems, and control, in 2018 and 2021, respectively, both from ETH Zurich, Switzerland. He is currently working toward the Ph.D. degree in robotics at the Robotic Systems Lab, ETH Zurich, under the supervision of Prof. Marco Hutter.

He is currently a Guest Lecturer at the University of Genoa, Italy. He held multiple teaching positions at ETH Zurich. During his studies he designed a mechanical watch prototype with novel complications, developed an autonomous delivery robot for the company Sevensense, and, as one of the inventors, initiated the Ascento robot project, which has recently turned into a startup (patent pending). His research interests include development of (wheeled–)legged robots, simulation of dynamical systems, and reinforcement learning for motion control.

Yvain de Viragh (Graduate Student Member, IEEE) received the B.Sc. degree in mechanical engineering and the M.Sc. degree in robotics, systems, and control, in 2016 and 2019, respectively, from ETH Zurich, Switzerland.

He has worked for the legged-robotics company ANYbotics, Zurich, Switzerland. According to Victor, he is "a free-spirited robotics, maths, and CNC enthusiast, aficionado, and researcher with techno-anarchist, multicultulinary [sic], and omnidisciplinary tendencies." According to himself, he is currently a calculus of variations, epistemology, and Wing Chun apprentice.

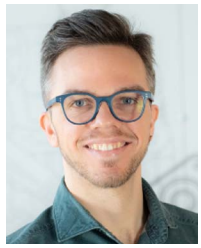


David Rohr received the B.Sc. degree in mechanical engineering and the M.Sc. degree in robotics, systems, and control, in 2016 and 2019, respectively, both from ETH Zurich, Switzerland. He is currently working toward the Ph.D. degree in robotics at the Autonomous Systems Lab, ETH Zurich, under the supervision of Prof. Roland Siegwart.

His research interests include modeling, online dynamics learning and control for hybrid aerial vehicles.

Roland Siegwart (Fellow, IEEE) is a Professor of autonomous mobile robots at ETH Zurich, Switzerland and the Founding Co-Director of the Technology Transfer Center, Wyss Zurich, Switzerland. From 1996 to 2006, he was a Professor at EPFL, Lausanne, Switzerland. He has held visiting positions at Stanford University, Stanford, CA, USA, and NASA Ames, Moffett Field, CA, USA. From 2010 to 2014, he was the Vice President of ETH Zurich. His research interests include the design, control, and navigation of flying and wheeled and walking robots operating in complex and highly dynamical environments.

Prof. Siegwart was a recipient of the IEEE RAS Pioneer Award and the IEEE RAS Inaba Technical Award. He is among the most cited scientists in robotics world-wide, a Co-Founder of more than half a dozen spin-off companies, and a strong Promoter of innovation and entrepreneurship in Switzerland. He is a Board Member of multiple high-tech companies.



Marco Tognon (Member, IEEE) received the M.Sc. degree in automation engineering from the University of Padua, Italy, in 2014, with a master thesis carried out at the Max Planck Institute for Biological Cybernetics, Tübingen, Germany, and the Ph.D. degree in robotics from the National Institute of Applied Sciences of Toulouse, France, in 2018, developing his thesis at the Laboratory for Analysis and Architecture of Systems, LAAS-CNRS, Toulouse, France.

Since 2022, he has been a Tenured Researcher with the Rainbow Team, INRIA Rennes, France. Before, he was a postdoctoral Researcher at the Autonomous Systems Lab, ETH Zurich. He has authored or coauthored more than 25 journal papers, and he is the coordinator of the ANR project AirHandyBot. His research interests include robotics, aerial physical interaction, multirobot systems, and human–robot physical interaction.

He is an Associate Editor for IEEE TRANSACTIONS ON ROBOTICS and an Area Chair for RSS2023. He is also junior Co-Chair for IEEE TC Aerial Robotics and Unmanned Aerial Vehicles.