Perceptive Locomotion Through Nonlinear Model-Predictive Control

Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter

***Abstract*—Dynamic locomotion in rough terrain requires accurate foot placement, collision avoidance, and planning of the underactuated dynamics of the system. Reliably optimizing for such motions and interactions in the presence of imperfect and often incomplete perceptive information is challenging. We present a complete perception, planning, and control pipeline, which can optimize motions for all degrees of freedom of the robot in real time. To mitigate the numerical challenges posed by the terrain, a sequence of convex inequality constraints is extracted as local approximations of foothold feasibility and embedded into an online model-predictive controller. Steppability classification, plane segmentation, and a signed distance field are precomputed per elevation map to minimize the computational effort during the optimization. A combination of multiple-shooting, real-time iteration, and a filter-based line search is used to solve the formulated problem reliably and at high rate. We validate the proposed method in scenarios with gaps, slopes, and stepping stones in simulation and experimentally on the ANYmal quadruped platform, resulting in state-of-the-art dynamic climbing.**

***Index Terms*—Legged locomotion, optimal control, terrain perception.**

I. Introduction

INSPIRED by nature, the field of legged robotics aims to enable the deployment of autonomous systems in rough and complex environments. Indeed, during the recent DARPA subterranean challenge, legged robots were widely adopted and highly successful [2], [3]. Still, complex terrains that require precise foot placements, e.g., negative obstacles and stepping stones, as shown in Fig. 1, remain difficult.

A key challenge lies in the fact that both the terrain and the system dynamics impose constraints on contact location, force, and timing. When taking a model-based approach, mature methods exist for perceptive locomotion with a slow static gait [4], [5], [6], [7], [8] and for blind dynamic locomotion that assumes flat terrain [9], [10], [11]. Learning-based controllers have recently shown the ability to generalize blind locomotion to challenging terrain with incredible robustness [12], [13], [14]. Still, tightly integrating perception to achieve coordinated and precise foot placement remains an active research problem.

In an effort to extend dynamic locomotion to uneven terrain, several methods have been proposed to augment foothold selection algorithms with perceptive information [15], [16], [17]. These approaches build on a strict hierarchy of first selecting footholds and optimizing torso motion afterward. This decomposition reduces the computational complexity but relies on hand-crafted coordination between the two modules. In addition, separating the legs from the torso optimization makes it difficult to consider kinematic limits and collision avoidance between limbs and terrain.

Trajectory optimization where torso and leg motions are jointly optimized has shown impressive results in simulation [18], [19], [20] and removes the need for engineered torso–foot coordination. Complex motions can be automatically discovered by including the entire terrain in the optimization. However, computation times are often too long for online deployment. In addition, due to the nonconvexity, nonlinearity, and discontinuity introduced by optimizing over arbitrary terrain, these methods can get stuck in poor local minima. Dedicated work on providing an initial guess is needed to find feasible motions reliably [21].

This article presents a planning and control framework that optimizes over all degrees of freedom of the robot, considers

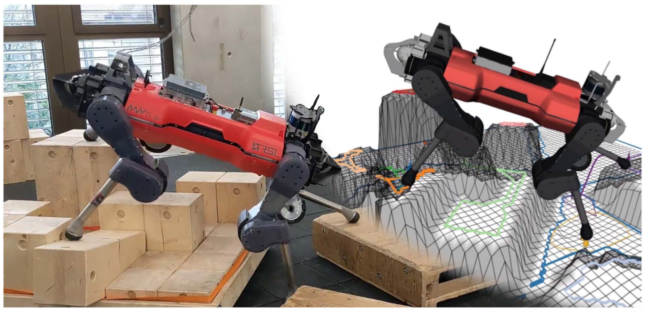


Fig. 1. ANYmal walking on uneven stepping stones. In the shown configuration, the top foothold is 60 cm above the lowest foothold. The top right visualizes the internal terrain representation used by the controller.

Manuscript received 17 August 2022; revised 6 February 2023; accepted 12 April 2023. Date of publication 29 May 2023; date of current version 4 October 2023. This work was supported in part by the Swiss National Science Foundation (SNSF) under Grant 188596, in part by the SNSF through the National Centre of Competence in Research Robotics (NCCR Robotics), and in part by the European Union's Horizon 2020 research and innovation program under Grant 780883. This paper was recommended for publication by Associate Editor C. Semini and Editor E. Yoshida upon evaluation of the reviewers' comments. *(Corresponding author: Ruben Grandia.)*

Ruben Grandia, Fabian Jenelten, Farbod Farshidian, and Marco Hutter are with the Department of Mechanical and Process Engineering, ETH Zurich, 8092 Zurich, Switzerland (e-mail: rgrandia@ethz.ch; fabian.jenelten@mavt.ethz.ch; farshidian@mavt.ethz.ch; mahutter@ethz.ch).

Shaohui Yang is with the Automatic Control Laboratory, École polytechnique fédérale de Lausanne, 1015 Lausanne, Switzerland (e-mail: shaohuiy@kth.se).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3275384.

Digital Object Identifier 10.1109/TRO.2023.3275384

collision avoidance with the terrain, and enables complex dynamic maneuvers in rough terrain. The method is centered around nonlinear model-predictive control (MPC) with a multiple-shooting discretization [22], [23]. However, in contrast to the aforementioned work, where the full terrain is integrated into the optimization, we get a handle on the numerical difficulty introduced by the terrain by exposing the terrain as a series of geometric primitives that approximate the local terrain. In this case, we use convex polygons as foot placement constraints, but different shapes can be used as long as they lead to well-posed constraints in the optimization. In addition, a signed distance field (SDF) is used for collision avoidance. We empirically demonstrate that such a strategy is an excellent tradeoff between giving freedom to the optimization to discover complex motions and the reliability with which we can solve the formulated problem.

A. Contributions

We present a novel approach to locomotion in challenging terrain where perceptive information needs to be considered and nontrivial motions are required. The complete perception, planning, and control pipeline contains the following contributions.

1) We propose an MPC architecture that enables simultaneous and real-time optimization of all degrees of freedom of the robot for dynamic motions across rough terrain. Perceptive information is encoded through a sequence of geometric primitives that capture local foothold constraints and a signed distance field used for collision avoidance.
2) We benchmark the combination of multiple-shooting transcription, sequential quadratic programming (SQP), and a custom filter-based line search procedure for fast and reliable online solutions to nonlinear optimal control problems for locomotion.

We provide a detailed description of the implemented MPC, its integration with whole-body and reactive control modules, and extensive experimental validation of the resulting locomotion controller. The MPC implementation is publicly available as part of the OCS2 toolbox[1] [24]. The implemented online segmentation of the elevation map and the efficient precomputation of a signed distance field are contributed to existing open-source repositories.[2,3]

B. Outline

An overview of the proposed method is given in Fig. 2. The perception pipeline at the top of the diagram runs at 20 Hz and is based on an elevation map constructed from point cloud information. For each map update, classification, segmentation, and other precomputation are performed to prepare for the high number of perceptive queries during motion optimization. At the core of the framework, we use nonlinear MPC at 100 Hz to plan a motion for all degrees of freedom and bring together

[1][Online]. Available: https://github.com/leggedrobotics/ocs2

[2][Online]. Available: https://github.com/leggedrobotics/elevation_mapping_cupy

[3][Online]. Available: https://github.com/ANYbotics/grid_map

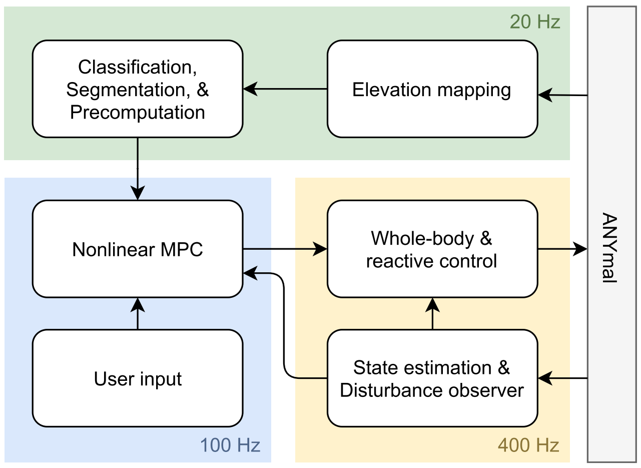

20 Hz
Classification, Segmentation, & Precomputation
Elevation mapping
Nonlinear MPC
Whole-body & reactive control
User input
State estimation & Disturbance observer
100 Hz
400 Hz
ANYmal

Fig. 2. Schematic overview of the proposed method together with the update rate of each component.

user input, perceptive information, and the measured state of the robot. Finally, state estimation, whole-body torque control, and reactive behaviors are executed at a rate of 400 Hz.

After a review of related work in Section II, this article is structured similarly to Fig. 2. First, we present the perception pipeline in Section III. Afterward, the formulated optimal control problem and the corresponding numerical optimization strategy are discussed in Sections IV and V, respectively. We introduce the motion execution layer in Section VI. The resulting method is evaluated on the quadrupedal robot ANYmal [25] (see Fig. 1) in Section VII. Finally, Section VIII concludes this article.

II. Related Work

A. Decomposing Locomotion

When assuming a quasi-static gait with a predetermined stepping sequence, the planning problem on rough terrain can be simplified and decomposed into individual contact transitions, as demonstrated in the early work on *LittleDog* [4], [26]. In a one-step-ahead fashion, one can check the next foothold for kinematic feasibility, feasibility w.r.t. the terrain, and the existence of a statically stable transition. This problem can be efficiently solved by sampling and checking candidate footholds [27]. Afterward, a collision-free swing leg trajectory to the desired foothold can be generated based on an SDF, for example, with CHOMP [28]. Fully onboard perception and control with such an approach were achieved by Fankhauser et al. [7]. Instead of one-step-ahead planning, a rapidly exploring random tree graph can be built to plan further ahead [5]. Sampling over templated foothold transitions achieves similar results [6], [29].

In this article, we turn our attention to dynamic gaits, where statically stable transitions between contact configurations are not available. In model-based approaches to dynamic perceptive locomotion, a distinction can be made between methods where the foothold locations are determined separately from the torso and those where the foothold locations and torso motions are jointly optimized.

Several methods in which footholds are selected before optimizing the torso motions, initially designed for flat terrain, have been adapted to traverse rough terrain [30], [31]. These methods typically employ some form of Raibert heuristic [32] to select the next foothold and adapt it based on perceptive information such as a traversability estimate [33]. The work of Bellicoso et al. [9] was extended by including a batch search for feasible footholds based on a given terrain map and foothold scoring [15]. Similarly, in [16], the foot placement is adapted based on visual information resulting in dynamic trotting and jumping motions. Magana et al. [34] proposed to train a convolutional neural network (CNN) to speed up the online evaluation of such a foothold adaptation pipeline. This CNN was combined with the MPC strategy in [11] to achieve perceptive locomotion in simulation [17]. In [35] and [36], a reinforcement learning (RL) policy has replaced the heuristic foothold selection.

However, since foothold locations are chosen before optimizing the torso motion, their effect on dynamic stability and kinematic feasibility is not directly considered, requiring additional heuristics to coordinate feet and torso motions to satisfy whole-body kinematics and dynamics. Moreover, it becomes hard to consider collisions of the leg with the terrain because the foothold is already fixed. In our approach, we use the same heuristics to find a suitable nominal foothold in the terrain. However, instead of fixing the foothold to that particular location, a region is extracted around the heuristic in which the foothold is allowed to be optimized.

The benefit of jointly optimizing torso and leg motions has been demonstrated in the field of trajectory optimization. One of the first demonstrations of simultaneous optimization of foot placement and a zero-moment point [37] trajectory was achieved by adding 2-D foot locations as decision variables to an MPC algorithm [38]. More recently, Kinodynamic [39], Centroidal [40], [41], [42], and full dynamics models [43], [44], [45] have been used for simultaneous optimization of 3-D foot locations and body motion. Alternatively, a single rigid-body dynamics (SRBD) model or other simplified torso models can be extended with decision variables for Cartesian foothold locations [19], [46]. Real-time capable methods have been proposed with the specification of leg motions on position [10], velocity [47], or acceleration level [48]. One challenge of this line of work is the computational complexity arising from the high-dimensional models, already in the case of locomotion on flat terrain. Our method also uses a high-dimensional model and falls in this category. A key consideration when extending the formulations with perceptive information has, thus, been to keep computation within real-time constraints.

Finally, several methods exist that additionally optimize gait timings or even the contact sequence together with the whole-body motion. This can be achieved through complementarity constraints [18], [20], [49], mixed-integer programming [50], [51], or by explicitly integrating contact models into the optimization [48], [52]. Alternatively, the duration of each contact phase can be included as a decision variable [19], [53] or found through bilevel optimization [54], [55]. However, such methods are prone to poor local optima, and reliably solving the optimization problems in real time remains challenging.

B. Terrain Representation

The use of an elevation map has a long-standing history in the field of legged robotics [56], and it is still an integral part of many perceptive locomotion controllers today. Approaches where footholds are selected based on a local search or sampling-based algorithm can directly operate on such a structure. However, more work is needed when integrating the terrain into a gradient-based optimization.

Winkler et al. [19] use an elevation map for both foot placement and collision avoidance. The splines representing the foot motion are constrained to start and end on the terrain with equality constraints. An inequality constraint is used to avoid the terrain in the middle of the swing phase. Ignoring the discontinuity and nonconvexity from the terrain makes this approach prone to poor local minima, motivating specialized initialization schemes [21] for this framework.

In [46], a graduated optimization scheme is used, where a first optimization is carried out over a smoothened version of the terrain. The solution of this first optimization is then used to initialize an optimization over the actual elevation map. In a similar spirit, Mordatch [18] considers a general 3-D environment and uses a soft-min operator to smoothen the closest point computation. A continuation scheme is used to gradually increase the difficulty of the problem over consecutive optimizations.

Deits and Tedrake [57] describe a planning approach over rough terrain based on mixed-integer quadratic programming (MIQP). Similar to [8], convex safe regions are extracted from the terrain, and footstep assignment to a region is formulated as a discrete decision. The foothold optimization is simplified because only convex safe regions are considered during planning. Furthermore, the implementation relied on the manual seeding of convex regions by a human operator. We follow the same philosophy of presenting the terrain as a convex region to the optimization. However, we remove the mixed-integer aspect by preselecting the convex region. The benefits are twofold: First, we do not require a global convex decomposition of the terrain, which is a hard problem in general [58], and, instead, only produce a local convex region centered around a nominal foothold. Second, the MIQP approach does not allow for nonlinear costs and dynamics, which limits the range of motions that can be expressed. We first explored the proposed terrain representation as part of our previous work [59], but relied on offline mapping, manual terrain segmentation, and did not yet consider terrain collisions. In [60], we applied this idea to wheeled-legged robots, but again relied on offline mapping and segmentation. Moreover, as discussed in the next section, in both [59] and [60], we used a different solver, which was found to be insufficient for the scenarios in this article.

C. Motion Optimization

For trajectory optimization, large-scale optimization packages, such as SNOPT [61] and IPOPT [62], are popular. They

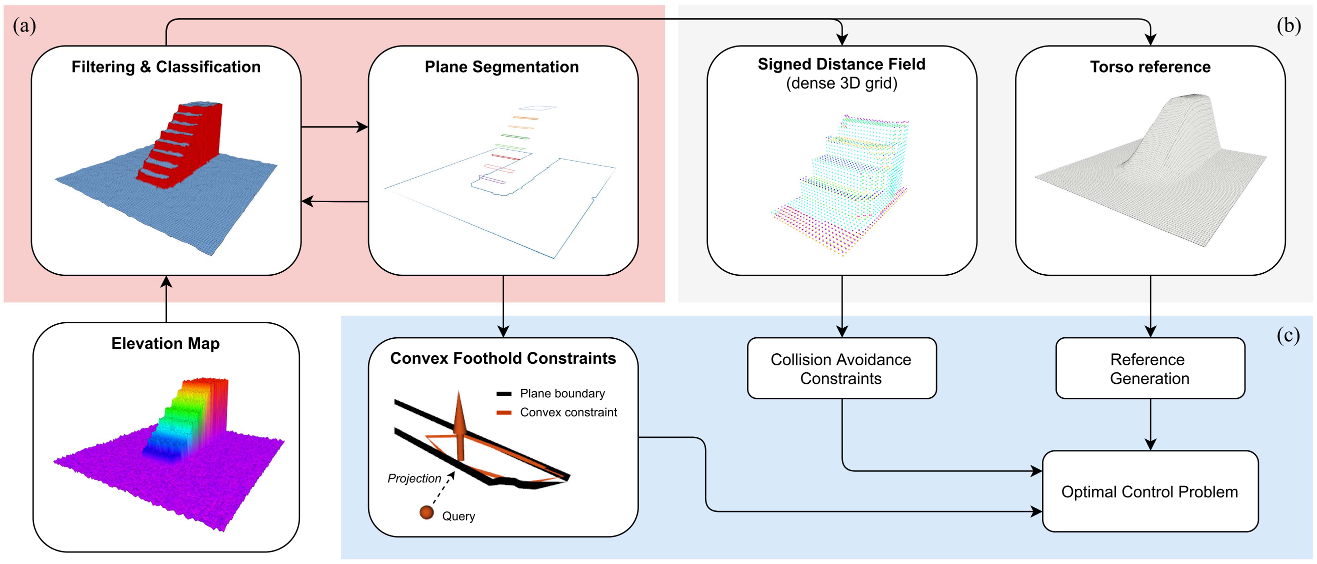


Fig. 3. Perception pipeline overview. (a) Elevation map is filtered and classified into steppable and nonsteppable cells (see Section III-A). All steppable areas are segmented into planes (see Section III-B). After segmentation, the steppability classification is refined. (b) Signed distance field (see Section III-C) and torso reference layer (see Section III-D) are precomputed to reduce the required computation time during optimization. (c) Convex foothold constraints in (21) are obtained from the plane segmentation. The signed distance field enables collision avoidance in (23), and the torso reference is used to generate height and orientation references (see Section IV-E).

are the workhorse for offline trajectory optimization in the work of Winkler [19], Dai [20], Mordatch [18], Posa [49], and Pardo [43]. These works show a great range of motions in simulation, but it typically takes minutes to hours to find a solution.

A different line of work uses specialized solvers that exploit the sparsity that arises from a sequential decision making process. Several variants of differential dynamic programming (DDP) [63] have been proposed in the context of robotic motion optimization, e.g., iLQR [64], [65], SLQ [39], and FDDP [45], [66].

With a slightly different view on the problem, the field of (nonlinear) MPC [23], [67] has specialized in solving successive optimal control problems under real-time constraints. See [68] for a comparison of state-of-the-art quadratic programming (QP) solvers that form the core of second-order optimization approaches to the nonlinear problem. For time-critical applications, the real-time iteration scheme can be used to trade optimality for lower computational demands [69]: In an SQP approach to the nonlinear problem, at each control instance, only a single QP optimization step is performed.

The current work was initially built on top of a solver in the first category [39]. However, a significant risk in classical DDP-based approaches is the need to perform a nonlinear system rollout along the entire horizon. Despite the use of a feedback policy, these forward rollouts can diverge, especially in the presence of underactuated dynamics. This same observation motivated Mastalli et al. to design FDDP to maintain *gaps* between shorter rollouts, resulting in a formulation that is equivalent to direct multiple-shooting formulations with only equality constraints [22], [66]. Giftthaler et al. [70] studied several combinations of iLQR and multiple shooting but did not yet consider constraints beyond system dynamics nor a line search procedure to determine the step size. Furthermore, experiments were limited to simple flat terrain walking.

We directly follow the multiple-shooting approach with a real-time iteration scheme and leverage the efficient structure exploiting QP solver HPIPM [71]. However, as also mentioned in both [66] and [70], one difficulty is posed in deciding a step size for nonlinear problems, where one now has to monitor both the violation of the system dynamics and minimization of the cost function. To prevent an arbitrary tradeoff through a merit function, we suggest using a filter-based line search instead [72], which allows a step to be accepted if it reduces either the objective function or the constraint violation. As we will demonstrate in the results section, these choices contribute to the robustness of the solver in challenging scenarios.

III. Terrain Perception and Segmentation

An overview of the perception pipeline and its relation to the MPC controller is provided in Fig. 3. The pipeline can be divided into three parts: (a) steppability classification and segmentation; (b) precomputation of the SDF and torso reference; and (c) integration into the optimal control problem.

The elevation map, represented as a 2.5-D grid [73] with a 4-cm resolution, is provided by the GPU-based implementation introduced in [74]. The subsequent map processing presented in this article runs on the CPU and is made available as part of that same open-source library. Both (a) and (b) are computed once per map and run at 20 Hz, asynchronously to the motion optimization in (c).

A. Filtering and Classification

The provided elevation map contains empty cells in occluded areas. As a first step, we perform *inpainting* by filling each cell with the minimum value found along the occlusion border. Afterward, a median filter is used to reduce noise and outliers in the map.

Steppability classification is performed by thresholding the local surface inclination and the local roughness estimated through the standard deviation [75]. Both quantities can be computed with a single pass through the considered neighborhood of size N:

$$\boldsymbol{\mu} = \frac{1}{N}\sum_i \mathbf{c}_i, \quad \mathbf{S} = \frac{1}{N}\sum_i \mathbf{c}_i\mathbf{c}_i^\top, \quad \boldsymbol{\Sigma} = \mathbf{S} - \boldsymbol{\mu}\boldsymbol{\mu}^\top, \tag{1}$$

where $\boldsymbol{\mu}$ and $\mathbf{S}$ are the first and second moments, and $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times 3}$ is the positive-semidefinite (p.s.d) covariance matrix of the cell positions $\mathbf{c}_i$. The variance in normal direction, σ_n^2, is then the smallest eigenvalue of $\boldsymbol{\Sigma}$, and the surface normal, $\mathbf{n}$, is the corresponding eigenvector. For steppability classification, we use a neighborhood of $N = 9$ and set a threshold of 2 cm on the standard deviation in normal direction and a maximum inclination of 35°, resulting in the following classification:

$$\text{steppability} = \begin{cases} 1, & \text{if } \sigma_n \leq 0.02 \text{ and } n_z \geq 0.82 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where n_z denotes the z-coordinate of the surface normal.

B. Plane Segmentation

After the initial classification, the plane segmentation starts by identifying continuous regions with the help of a connected component labeling [76]. For each connected region of cells, we compute again the covariance as in (1), where N is now the number of cells in the connected region, and accept the region as a plane based on the following criteria:

$$\text{planarity} = \begin{cases} 1, & \text{if } \sigma_n \leq 0.025, n_z \geq 0.87, \text{ and } N \geq 4 \\ 0, & \text{otherwise} \end{cases}. \tag{3}$$

Notice that here we loosen the bound on the standard deviation to 2.5 cm, tighten the bound on the inclination to 30 °, and add the constraint that at least four cells form a region. If the planarity condition is met, the surface normal and mean of the points define the plane.

If a region fails the planarity condition, we trigger RANSAC [77] on that subset of the data. The same criteria in (3) are used to find smaller planes within the connected region. After the algorithm terminates, all cells that have not been included in any plane have their steppability updated and set to 0.

At this point, we have a set of plane parameters with connected regions of the map assigned to them. For each of these regions, we now extract a 2-D contour from the elevation map [78] and project it along the z-axis to the plane to define the boundary in the frame of the plane. It is important to consider that regions can have holes, for example, when a pit or an obstacle is located in the middle of an open floor. The boundary of each segmented region is, therefore, represented by an outer polygon together with a set of polygons that trace enclosed holes. See Fig. 4 for an illustrative example of such a segmented region and the local convex approximations it permits. Finally, if the particular region

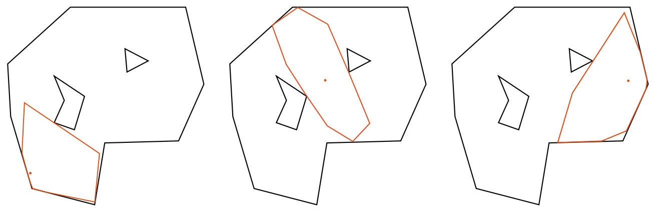


Fig. 4. Example of a segmented region represented by a nonconvex outer polygon and two nonoverlapping holes (drawn in black). Three different local convex approximations (drawn in orange) are shown that are found around query points with the iterative algorithm described in Section IV-F.

allows, we shrink the boundary inward (and holes outward) to provide a safety margin. If the inscribed area is not large enough, the plane boundary is accepted without margin. In this way, we obtain a margin where there is enough space to do so, but at the same time, we do not reject small stepping stones, which might be crucial in certain scenarios.

C. Signed Distance Field

Before computing the SDF, we take advantage of the classification between terrain that will be potentially stepped on and terrain that will not be stepped on. To all cells that are nonsteppable, we add a vertical margin of 2 cm and dilate the elevation by one cell. The latter effectively horizontally inflates all nonsteppable areas by the map resolution. This procedure corrects for the problem that edges tend to be underestimated in the provided elevation map.

We use a dense 3-D voxel grid, where each voxel contains the value and 3-D gradient. The previous motion plan is used to determine the 3-D volume where distance information is needed. This volume is a bounding box that contains all collision bodies of the last available plan with a margin of 25 cm. This way, the size and shape of the SDF grid dynamically scales with the motion that is executed. Storing both value and derivative as proposed in [79] allows for efficient interpolation during optimization. However, in contrast to [79], where values and gradients are cached after the first call, we opt to precompute the full voxel grid to reduce the computation time during optimization as much as possible.

This is possible by taking advantage of the extra structure that the 2.5-D representation provides. A detailed description of how the SDF can be efficiently computed from an elevation map is given in Appendix A.

D. Torso Reference Map

With user input defined as horizontal velocity and an angular rate along the z-direction, it is the responsibility of the controller to decide on the height and orientation of the torso. We would like the torso pose to be positioned in such a way that suitable footholds are in reach for all of the feet. We, therefore, create a layer that is a smooth interpolation of all steppable regions, as described in [46]. The use of this layer to generate a torso height and orientation reference is presented in Section IV-E.

IV. Motion Planning

In this section, we describe the nonlinear MPC formulation. In particular, we set out to define all components in the following nonlinear optimal control problem:

$$\begin{aligned} \underset{\mathbf{u}(\cdot)}{\text{minimize}} \quad & \Phi(\mathbf{x}(T)) + \int_0^T L(\mathbf{x}(t), \mathbf{u}(t), t) dt && \text{(4a)} \\ \text{subject to} \quad & \mathbf{x}(0) = \hat{\mathbf{x}} && \text{(4b)} \\ & \dot{\mathbf{x}} = \mathbf{f}^c(\mathbf{x}, \mathbf{u}, t) && \text{(4c)} \\ & \mathbf{g}(\mathbf{x}, \mathbf{u}, t) = \mathbf{0} && \text{(4d)} \end{aligned}$$

where $\mathbf{x}(t)$ and $\mathbf{u}(t)$ are the state and the input at time t, respectively, and $\hat{\mathbf{x}}$ is the current estimated state. The term $L(\cdot)$ is a time-varying running cost, and $\Phi(\cdot)$ is the cost at the terminal state $\mathbf{x}(T)$. The goal is to find a control signal that minimizes this cost subject to the initial condition, $\mathbf{x}_0$, system dynamics, $\mathbf{f}^c(\cdot)$, and equality constraints, $\mathbf{g}(\cdot)$. Inequality constraints are all handled through penalty functions and will be defined as part of the cost function in Section IV-F.

A. Robot Definition

We define the generalized coordinates and velocities as

$$\mathbf{q} = \left[\boldsymbol{\theta}_B^\top, \mathbf{p}_B^\top, \mathbf{q}_j^\top\right]^\top, \quad \dot{\mathbf{q}} = \left[\boldsymbol{\omega}_B^\top, \mathbf{v}_B^\top, \dot{\mathbf{q}}_j^\top\right]^\top \tag{5}$$

where $\boldsymbol{\theta}_B \in \mathbb{R}^3$ is the orientation of the base frame, $\mathcal{F}_B$, in Euler angles, and $\mathbf{p}_B \in \mathbb{R}^3$ is the position of the base in the world frame, $\mathcal{F}_W$. $\boldsymbol{\omega}_B \in \mathbb{R}^3$ and $\mathbf{v}_B \in \mathbb{R}^3$ are the angular rate and linear velocity of the base in the body frame $\mathcal{F}_B$, respectively. Joint positions and velocities are given by $\mathbf{q}_j \in \mathbb{R}^{12}$ and $\dot{\mathbf{q}}_j \in \mathbb{R}^{12}$, respectively. The collection of all contact forces is denoted by $\boldsymbol{\lambda} \in \mathbb{R}^{12}$. When referring to these quantities per leg, we will use a subscript i, e.g., $\mathbf{q}_i \in \mathbb{R}^3$ or $\boldsymbol{\lambda}_i \in \mathbb{R}^3$. All subscripts for legs in contact are contained in the set $\mathcal{C}$. A graphical illustration of the robot together with the defined coordinate frames is provided in Fig. 5.

B. Torso Dynamics

To derive the torso dynamics used in this article, consider the full rigid-body dynamics of the robot

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^\top \boldsymbol{\tau} + \boldsymbol{\tau}^{\text{dist}} + \sum_{i \in \mathcal{C}} \mathbf{J}_i^\top(\mathbf{q})\boldsymbol{\lambda}_i \tag{6}$$

with inertia matrix $\mathbf{M} : \mathbb{R}^{18} \to \mathbb{R}^{18\times18}$, generalized accelerations $\ddot{\mathbf{q}} \in \mathbb{R}^{18}$, and nonlinear terms $\mathbf{n} : \mathbb{R}^{18} \times \mathbb{R}^{18} \to \mathbb{R}^{18}$ on the left-hand side. The right-hand side contains the selection matrix $\mathbf{S} = [\mathbf{0}_{12\times6}, \mathbf{I}_{12\times12}] \in \mathbb{R}^{12\times18}$, actuation torques $\boldsymbol{\tau} \in \mathbb{R}^{12}$, disturbance forces $\boldsymbol{\tau}^{\text{dist}} \in \mathbb{R}^{18}$, contact Jacobians $\mathbf{J}_i : \mathbb{R}^{18} \to \mathbb{R}^{3\times18}$, and contact forces $\boldsymbol{\lambda}_i \in \mathbb{R}^3$.

For these equations of motion, it is well known that for an articulated system, the underactuated top six rows are of main interest for motion planning [53]. These so-called centroidal dynamics govern the range of motion that can be achieved [40], [80]. Solving the centroidal dynamics for base acceleration gives

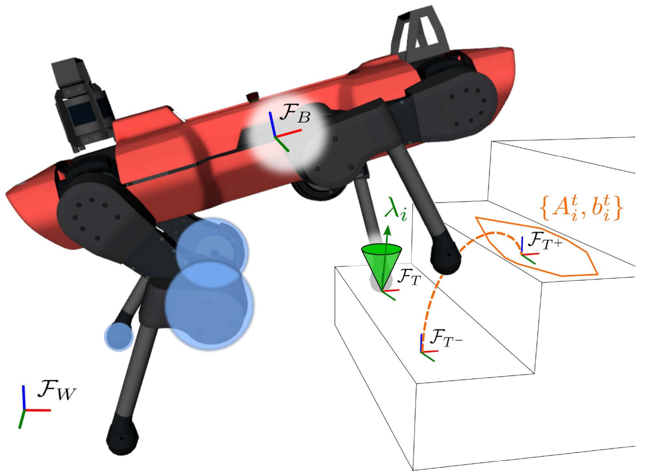


Fig. 5. Overview of the coordinates frames and constraints used in the definition of the MPC problem. On the front left foot, a friction cone is shown, defined in the terrain frame $\mathcal{F}_T$. On the right front foot, a swing reference trajectory is drawn between the liftoff frame $\mathcal{F}_{T-}$ and touchdown frame $\mathcal{F}_{T+}$. Foot placement constraints are defined as a set of half-spaces in the touchdown frame. Stance legs have collision bodies at the knee, as illustrated on the right hind leg, while swing legs have collision bodies on both the foot and the knee, as shown on the left hind leg.

$$\begin{aligned} \begin{bmatrix} \dot{\boldsymbol{\omega}}_B \\ \dot{\mathbf{v}}_B \end{bmatrix} &= \mathbf{M}_B^{-1}\left(\boldsymbol{\tau}_B^{\text{dist}} - \mathbf{M}_{Bj}\ddot{\mathbf{q}}_j - \mathbf{n}_B + \sum_{i\in\mathcal{C}} \mathbf{J}_{B,i}^\top \boldsymbol{\lambda}_i\right) && (7) \\ &= \mathbf{f}_B(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_j, \boldsymbol{\lambda}, \boldsymbol{\tau}_B^{\text{dist}}) && (8) \end{aligned}$$

where $\mathbf{M}_B(\mathbf{q}) \in \mathbb{R}^{6\times6}$ is the net composite rigid-body inertia matrix at the top left of $\mathbf{M}(\mathbf{q})$, and $\mathbf{M}_{Bj}(\mathbf{q}) \in \mathbb{R}^{6\times12}$ is the top right block that encodes inertial coupling between the legs and base. The other terms with subscript B correspond to the top six rows of the same terms in (6).

To simplify the torso dynamics, we evaluate this function with zero inertial coupling forces from the joints, i.e., $\mathbf{M}_{Bj}\ddot{\mathbf{q}}_j = \mathbf{0}$. This simplification allows us to consider the legs only on velocity level and removes joint accelerations from the formulation. From here, further simplifications would be possible. Evaluating the function at a nominal joint configuration and zero joint velocity creates a constant inertia matrix and gives rise to the commonly used single rigid-body assumption. While this assumption is appropriate on flat terrain, the joints move far away from their nominal configuration in this work, creating a significant shift in mass distribution and center of mass location.

C. Input Loop Shaping

The bandwidth limitations of the series elastic actuators used in ANYmal pose an additional constraint on the set of motions that are feasible on hardware. Instead of trying to accurately model these actuator dynamics, we use a frequency-dependent cost function to penalize high-frequency content in the contact forces and joint velocity signals [81]. For completeness, we present here the resulting system augmentation in the time domain

$$\dot{\mathbf{s}}_\lambda = \mathbf{A}_\lambda \mathbf{s}_\lambda + \mathbf{B}_\lambda \boldsymbol{\nu}_\lambda, \qquad \dot{\mathbf{s}}_j = \mathbf{A}_j \mathbf{s}_j + \mathbf{B}_j \boldsymbol{\nu}_j$$
$$\boldsymbol{\lambda} = \mathbf{C}_\lambda \mathbf{s}_\lambda + \mathbf{D}_\lambda \boldsymbol{\nu}_\lambda, \qquad \dot{\mathbf{q}}_j = \mathbf{C}_j \mathbf{s}_j + \mathbf{D}_j \boldsymbol{\nu}_j \tag{9}$$

where $\mathbf{s}_\lambda$ and $\mathbf{s}_j$ are additional states, and $\boldsymbol{\nu}_\lambda$ and $\boldsymbol{\nu}_j$ are auxiliary inputs, associated with contact forces and joint velocities, respectively. When the filters ($\boldsymbol{\nu}_\lambda \rightarrow \boldsymbol{\lambda}$ and $\boldsymbol{\nu}_j \rightarrow \dot{\mathbf{q}}_j$) are low-pass filters, penalizing the auxiliary input is equivalent to penalizing high-frequency content in $\boldsymbol{\lambda}$ and $\dot{\mathbf{q}}_j$.

An extreme case is obtained when choosing $\mathbf{A}_\lambda = \mathbf{D}_\lambda = \mathbf{0}$ and $\mathbf{B}_\lambda = \mathbf{C}_\lambda = \mathbf{I}$, in which case the auxiliary input becomes the derivative, $\dot{\boldsymbol{\lambda}}$. This reduces to the common system augmentation technique that allows the penalization of input rates [23].

In our case, we allow some direct control ($\mathbf{D} \neq \mathbf{0}$) and select $\mathbf{A}_\lambda = \mathbf{A}_j = \mathbf{0}$, $\mathbf{B}_\lambda = \mathbf{B}_j = \mathbf{I}$, $\mathbf{C}_\lambda = \frac{100}{4}\mathbf{I}$, $\mathbf{C}_j = \frac{50}{3}\mathbf{I}$, $\mathbf{D}_\lambda = \frac{1}{4}\mathbf{I}$, and $\mathbf{D}_j = \frac{1}{3}\mathbf{I}$. This corresponds to a progressive increase in cost up to a frequency of 100 rad s^{-1} for $\boldsymbol{\lambda}$ and up to 50 rad s^{-1} for $\dot{\mathbf{q}}_j$, where high-frequency components have their cost increased by a factor of 4 and 3, respectively.

D. System Dynamics

We are now ready to define the state vector $\mathbf{x} \in \mathbb{R}^{48}$ and input vector $\mathbf{u} \in \mathbb{R}^{24}$ used during motion optimization

$$\mathbf{x} = \left[\boldsymbol{\theta}_B^\top, \mathbf{p}_B^\top, \boldsymbol{\omega}_B^\top, \mathbf{v}_B^\top, \mathbf{q}_j^\top, \mathbf{s}_\lambda^\top, \mathbf{s}_j^\top\right]^\top, \quad \mathbf{u} = \left[\boldsymbol{\nu}_\lambda^\top, \boldsymbol{\nu}_j^\top\right]^\top. \tag{10}$$

Putting together the robot dynamics from Section IV-B and system augmentation described in Section IV-C gives the continuous-time MPC model $\dot{\mathbf{x}} = \mathbf{f}^c(\mathbf{x}, \mathbf{u}, t)$

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \boldsymbol{\theta}_B \\ \mathbf{p}_B \\ \boldsymbol{\omega}_B \\ \mathbf{v}_B \\ \mathbf{q}_j \\ \mathbf{s}_\lambda \\ \mathbf{s}_j \end{bmatrix} = \begin{bmatrix} \mathbf{T}(\boldsymbol{\theta}_B)\boldsymbol{\omega}_B \\ \mathbf{R}_B(\boldsymbol{\theta}_B)\,\mathbf{v}_B \\ \mathbf{f}_B(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{0}, \mathbf{C}_\lambda \mathbf{s}_\lambda + \mathbf{D}_\lambda \boldsymbol{\nu}_\lambda, \boldsymbol{\tau}_B^{\text{dist}}) \\ \mathbf{C}_j \mathbf{s}_j + \mathbf{D}_j \boldsymbol{\nu}_j \\ \mathbf{A}_\lambda \mathbf{s}_\lambda + \mathbf{B}_\lambda \boldsymbol{\nu}_\lambda \\ \mathbf{A}_j \mathbf{s}_j + \mathbf{B}_j \boldsymbol{\nu}_j \end{bmatrix} \tag{11}$$

where $\mathbf{T}(\boldsymbol{\theta}_B) : \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times 3}$ provides the conversion between body angular velocity and Euler angle derivatives, and $\mathbf{R}_B(\boldsymbol{\theta}_B) : \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times 3}$ provides the body to world rotation matrix. The disturbance wrench $\boldsymbol{\tau}_B^{\text{dist}}$ is considered a parameter and is assumed constant over the MPC horizon.

E. Reference Generation

The user commands 2-D linear velocities and an angular rate in the horizontal plane, as well as a desired gait pattern. A full motion and contact force reference is generated to encode these user commands and additional motion preferences into the cost function defined in Section IV-F. This process is carried out before every MPC iteration.

As a first step, assuming a constant input along the horizon, a 2-D base reference position and heading direction are extrapolated in the world frame. At each point in time, the 2-D pose is converted to a 2-D position for each hip. The smoothened elevation map, i.e., the *torso reference* layer shown in Fig. 3, is interpolated at the 2-D hip location. The interpolated elevation in addition to a desired nominal height, h_{nom}, gives a 3-D reference position for each hip. A least-squares fit through the four hip positions gives the six-degree-of-freedom base reference.

The extracted base reference and the desired gait pattern are used to derive nominal foothold locations. Here, we use the common heuristic that the nominal foothold is located below the hip, in gravity-aligned direction, at the middle of the contact phase [32]. In addition, for the first upcoming foothold, a feedback on the measured velocity is added

$$\mathbf{p}_{i,\text{nom}} = \mathbf{p}_{i,\text{hip,nom}} + \sqrt{\frac{h_{\text{nom}}}{g}}(\mathbf{v}_{B,\text{meas}} - \mathbf{v}_{B,\text{com}}) \tag{12}$$

where $\mathbf{p}_{i,\text{nom}} \in \mathbb{R}^3$ is the nominal foothold, $\mathbf{p}_{i,\text{hip,nom}} \in \mathbb{R}^3$ is the nominal foothold location directly below the hip, and g is the gravitational constant. $\mathbf{v}_{B,\text{meas}}$ and $\mathbf{v}_{B,\text{com}}$ are measured and commanded base velocity, respectively.

With the nominal foothold locations known, the plane segmentation defined in Section III-B is used to adapt the nominal foothold locations to the perceived terrain. Each foothold is projected onto the plane that is closest and within kinematic limits. Concretely, we pick the reference foothold, $\mathbf{p}_{i,\text{proj}}$, according to

$$\underset{\mathbf{p}_{i,\text{proj}} \in \Pi(\mathbf{p}_{i,\text{nom}})}{\text{argmin}} \; \|\mathbf{p}_{i,\text{nom}} - \mathbf{p}_{i,\text{proj}}\|_2^2 + w_{\text{kin}} f_{\text{kin}}(\mathbf{p}_{i,\text{proj}}) \tag{13}$$

where $\Pi(\mathbf{p}_{i,\text{nom}})$ is a set of candidate points. For each segmented plane, we take the point within that region that is closest to the nominal foothold as a candidate. The term f_{kin} is a kinematic penalty with weight w_{kin} that penalizes the point if the leg extension at liftoff or touchdown is beyond a threshold and if the foothold crosses over to the opposite side of the body. Essentially, this is a simplified version of the foothold batch search algorithm presented in [15], which searches over cells of the map instead of presegmented planes.

After computing all projected footholds, heuristic swing trajectories are computed with two quintic splines: from liftoff to apex and apex to touchdown. The spline is constrained by a desired liftoff and touchdown velocity, and an apex location is selected in such a way that the trajectory clears the highest terrain point between the footholds. Inverse kinematics is used to derive joint position references corresponding to the base and feet references. Finally, contact forces references are computed by dividing the total weight of the robot equally among all feet that are in contact. Joint velocity references are set to zero.

F. Cost and Soft Inequality Constraints

The cost function (4a) is built out of several components. The running cost $L(\mathbf{x}, \mathbf{u}, t)$ can be split into tracking costs $L_{\boldsymbol{\epsilon}}$, loop-shaping costs $L_{\boldsymbol{\nu}}$, and penalty costs $L_{\mathcal{B}}$

$$L = L_{\boldsymbol{\epsilon}} + L_{\boldsymbol{\nu}} + L_{\mathcal{B}}. \tag{14}$$

The motion tracking cost is used to follow the reference trajectory defined in Section IV-E. Tracking errors are defined

TABLE I
MOTION TRACKING WEIGHTS

Term	Weights
$\log(\mathbf{R}_B\mathbf{R}_{B,\text{ref}}^\top)^\vee$	$(100.0, 300.0, 300.0)$
$\mathbf{p}_B - \mathbf{p}_{B,\text{ref}}$	$(1000.0, 1000.0, 1500.0)$
$\boldsymbol{\omega}_B - \boldsymbol{\omega}_{B,\text{ref}}$	$(10.0, 30.0, 30.0)$
$\mathbf{v}_B - \mathbf{v}_{B,\text{ref}}$	$(15.0, 15.0, 30.0)$
$\mathbf{q}_i - \mathbf{q}_{i,\text{ref}}$	$(2.0, 2.0, 1.0)$
$\dot{\mathbf{q}}_i - \dot{\mathbf{q}}_{i,\text{ref}}$	$(0.02, 0.02, 0.01)$
$\mathbf{p}_i - \mathbf{p}_{i,\text{ref}}$	$(30.0, 30.0, 30.0)$
$\mathbf{v}_i - \mathbf{v}_{i,\text{ref}}$	$(15.0, 15.0, 15.0)$
$\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i,\text{ref}}$	$(0.001, 0.001, 0.001)$

for the base, $\boldsymbol{\epsilon}_B$, and for each foot, $\boldsymbol{\epsilon}_i$, as follows:

$$\boldsymbol{\epsilon}_B = \begin{bmatrix} \log(\mathbf{R}_B\mathbf{R}_{B,\text{ref}}^\top)^\vee \\ \mathbf{p}_B - \mathbf{p}_{B,\text{ref}} \\ \boldsymbol{\omega}_B - \boldsymbol{\omega}_{B,\text{ref}} \\ \mathbf{v}_B - \mathbf{v}_{B,\text{ref}} \end{bmatrix}, \ \boldsymbol{\epsilon}_i = \begin{bmatrix} \mathbf{q}_i - \mathbf{q}_{i,\text{ref}} \\ \dot{\mathbf{q}}_i - \dot{\mathbf{q}}_{i,\text{ref}} \\ \mathbf{p}_i - \mathbf{p}_{i,\text{ref}} \\ \mathbf{v}_i - \mathbf{v}_{i,\text{ref}} \\ \boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i,\text{ref}} \end{bmatrix} \tag{15}$$

where $\log(\mathbf{R}_B\mathbf{R}_{B,\text{ref}}^\top)^\vee$ is the logarithmic map of the orientation error, represented as a 3-D rotation vector, and $\mathbf{p}_i$ and $\mathbf{v}_i$ are the foot position and velocity in world frame, respectively. Together with diagonal positive-definite weight matrices $\mathbf{W}_B$ and $\mathbf{W}_i$, for which the individual elements are listed in Table I, these errors form the following nonlinear least-squares cost:

$$L_{\boldsymbol{\epsilon}} = \frac{1}{2}\|\boldsymbol{\epsilon}_B\|^2_{\mathbf{W}_B} + \sum_{i=1}^{4}\frac{1}{2}\|\boldsymbol{\epsilon}_i\|^2_{\mathbf{W}_i}. \tag{16}$$

As discussed in Section IV-C, high-frequency content in joint velocities and contact forces are penalized through a cost on the corresponding auxiliary input. This cost is a simple quadratic cost

$$L_{\boldsymbol{\nu}} = \frac{1}{2}\boldsymbol{\nu}_\lambda^\top\mathbf{R}_\lambda\boldsymbol{\nu}_\lambda + \frac{1}{2}\boldsymbol{\nu}_j^\top\mathbf{R}_j\boldsymbol{\nu}_j \tag{17}$$

where $\mathbf{R}_\lambda$ and $\mathbf{R}_j$ are constant p.s.d weight matrices. To obtain an appropriate scaling and avoid further manual tuning, these matrices are obtained from the quadratic approximation of the motion tracking cost (16), with respect to $\boldsymbol{\lambda}$ and $\dot{\mathbf{q}}_j$, respectively, at the nominal stance configuration of the robot.

All inequality constraints are handled through the penalty cost. In this article, we use relaxed barrier functions [82], [83]. This penalty function is defined as a log barrier on the interior of the feasible space and switches to a quadratic function at a distance δ from the constraint boundary

$$\mathcal{B}(h) = \begin{cases} -\mu\ln(h), & h \geq \delta \\ \frac{\mu}{2}\left(\left(\frac{h-2\delta}{\delta}\right)^2 - 1\right) - \mu\ln(\delta), & h < \delta \end{cases}. \tag{18}$$

The penalty is taken elementwise for vector-valued inequality constraints. The sum of all penalties is given as follows:

$$L_{\mathcal{B}} = \sum_{i=1}^{4}\mathcal{B}_j\left(\mathbf{h}_i^j\right) + \sum_{i\in\mathcal{C}}\mathcal{B}_t\left(\mathbf{h}_i^t\right) + \mathcal{B}_\lambda\left(h_i^\lambda\right) + \sum_{c\in\mathcal{D}}\mathcal{B}_d\left(h_c^d\right) \tag{19}$$

with joint limit constraints $\mathbf{h}_i^j$ for all legs, foot placement, and friction cone constraints, $\mathbf{h}_i^t$ and h_i^λ, for legs in contact, and collision avoidance constraints h_c^d for all bodies in a set $\mathcal{D}$.

The joint limit constraints contain upper $\{\overline{\mathbf{q}}_j, \overline{\dot{\mathbf{q}}}_j, \overline{\boldsymbol{\tau}}\}$ and lower bounds $\{\underline{\mathbf{q}}_j, \underline{\dot{\mathbf{q}}}_j, \underline{\boldsymbol{\tau}}\}$ for positions, velocities, and torques

$$\mathbf{h}_i^j = \begin{bmatrix} \overline{\mathbf{q}}_j - \mathbf{q}_j \\ \mathbf{q}_j - \underline{\mathbf{q}}_j \\ \overline{\dot{\mathbf{q}}}_j - \dot{\mathbf{q}}_j \\ \dot{\mathbf{q}}_j - \underline{\dot{\mathbf{q}}}_j \\ \overline{\boldsymbol{\tau}} - \boldsymbol{\tau} \\ \boldsymbol{\tau} - \underline{\boldsymbol{\tau}} \end{bmatrix} \geq \mathbf{0} \tag{20}$$

where we approximate the joint torques by considering a static equilibrium in each leg, i.e., $\boldsymbol{\tau}_i = \mathbf{J}_{j,i}^\top\boldsymbol{\lambda}_i$.

The foot placement constraint is a set of linear inequality constraints in task space

$$\mathbf{h}_i^t = \mathbf{A}_i^t \cdot \mathbf{p}_i + \mathbf{b}_i^t \geq \mathbf{0} \tag{21}$$

where $\mathbf{A}_i^t \in \mathbb{R}^{m\times 3}$ and $\mathbf{b}_i^t \in \mathbb{R}^m$ define m half-space constraints in 3-D. Each half-space is defined as the plane spanned by an edge of the 2-D polygon and the surface normal of the touchdown terrain $\mathcal{F}_{T+}$. The polygon is obtained by initializing all m vertices at the reference foothold derived in Section IV-E and iteratively displacing them outward. Each vertex is displaced in a round-robin fashion until it reaches the boundary of the segmented region or until further movement would cause the polygon to become nonconvex. Similar to [84], we have favored the low computational complexity of an iterative scheme over an exact approach of obtaining a convex inner approximation. The first set of extracted constraints remains unaltered for a foot that is in the second half of the swing phase to prevent last-minute jumps in constraints.

The friction cone constraint is implemented as

$$h_i^\lambda = \mu_c F_z - \sqrt{F_x^2 + F_y^2 + \epsilon^2} \geq 0 \tag{22}$$

with $[F_x, F_y, F_z]^\top = \mathbf{R}_T^\top\mathbf{R}_B\boldsymbol{\lambda}_i$, defining the forces in the local terrain frame. μ_c is the friction coefficient, and $\epsilon > 0$ is a parameter that ensures a continuous derivative at $\boldsymbol{\lambda}_i = \mathbf{0}$ and, at the same time, creates a safety margin [85].

The collision avoidance constraint is given by the evaluation of the SDF at the center of a collision sphere, $\mathbf{p}_c$, together with the required distance given by the radius, r_c, and a shaping function $d_{\text{min}}(t)$

$$h_c^d = d^{\text{SDF}}(\mathbf{p}_c) - r_c - d_{\text{min}}(t) \geq 0. \tag{23}$$

The primary use of the shaping function is to relax the constraint if a foot starts a swing phase from below the map. To avoid the robot using maximum velocity to escape the collision, we provide smooth guidance back to free space with a cubic spline trajectory. This happens when the perceived terrain is higher than the actual terrain, for example, in case of a soft terrain like vegetation and snow, or simply because of drift and errors in the estimated map. The collision set $\mathcal{D}$ contains collision bodies for

Algorithm 1: Real-Time Iteration Multiple-Shooting MPC.

```
1: Given: previous solution $\mathbf{w}_i$
2: Discretize the continuous problem to the form of (27)
3: Compute the linear–quadratic approximation (30)
4: Compute the equality constraint projection (34)
5: $\delta\tilde{\mathbf{w}} \leftarrow$ Solve the projected QP subproblem (35)
6: $\delta\mathbf{w} \leftarrow \mathbf{P}\delta\tilde{\mathbf{w}} + \mathbf{p}$, back substitution using (33)
7: $\mathbf{w}_{i+1} \leftarrow$ LineSearch($\mathbf{w}_i$, $\delta\mathbf{w}$), (Algorithm 2)
```

all knees and for all feet that are in swing phase, as visualized on the hind legs in Fig. 5.

Finally, we use a quadratic cost as the terminal cost in (4a). To approximate the infinite horizon cost incurred after the finite horizon length, we solve a linear–quadratic regulator (LQR) problem for the linear approximation of the MPC model and quadratic approximation of the intermediate costs around the nominal stance configuration of the robot. The Riccati matrix $\mathbf{S}_{\text{LQR}}$ of the cost-to-go is used to define the quadratic cost around the reference state

$$\Phi(\mathbf{x}) = \frac{1}{2}\left(\mathbf{x} - \mathbf{x}_{\text{ref}}(T)\right)^\top \mathbf{S}_{\text{LQR}}\left(\mathbf{x} - \mathbf{x}_{\text{ref}}(T)\right). \tag{24}$$

G. Equality Constraints

For each foot in swing phase, the contact forces are required to be zero

$$\boldsymbol{\lambda}_i = \mathbf{0} \qquad \forall i \notin \mathcal{C}. \tag{25}$$

In addition, for each foot in contact, the end-effector velocity is constrained to be zero. For swing phases, the reference trajectory is enforced only in the normal direction. This ensures that the foot lifts off and touches down with a specified velocity while leaving complete freedom of foot placement in the tangential direction

$$\begin{cases} \mathbf{v}_i = \mathbf{0}, & \text{if } i \in \mathcal{C} \\ \mathbf{n}^\top(t)\left(\mathbf{v}_i - \mathbf{v}_{i,\text{ref}} + k_p(\mathbf{p}_i - \mathbf{p}_{i,\text{ref}})\right) = 0, & \text{if } i \notin \mathcal{C} \end{cases}$$

The surface normal, $\mathbf{n}(t)$, is interpolated over time since liftoff and touchdown terrain can have a different orientation.

V. Numerical Optimization

We consider a direct multiple-shooting approach to transforming the continuous optimal control problem into a finite-dimensional nonlinear program (NLP) [22]. Since MPC computes control inputs over a receding horizon, successive instances of (27) are similar and can be efficiently warm-started when taking an SQP approach by shifting the previous solution. For new parts of the shifted horizon, for which no initial guess exists, we repeat the final state of the previous solution and initialize the inputs with the references generated in Section IV-E. In addition, we follow the real-time iteration scheme where only one SQP step is performed per MPC update [86]. In this way, the solution is improved across consecutive instances of the problem, rather than iterating until convergence for each problem.

As an overview of the approach described in the following sections, a pseudocode is provided in Algorithm 1, referring to the relevant equations used at each step. Except for the solution of the QP in line 5, all steps of the algorithm are parallelized across the shooting intervals. The QP is solved using HPIPM [71].

A. Discretization

The continuous control signal $\mathbf{u}(t)$ is parameterized over subintervals of the prediction horizon $[t, t+T]$ to obtain a finite-dimensional decision problem. This creates a grid of nodes $k \in \{0, \ldots, N\}$ defining control times t_k separated by intervals of duration $\delta t \approx T/(N-1)$. Around gait transitions, δt is slightly shortened or extended such that a node is exactly at the gait transition.

In this article, we consider a piecewise constant, or zero-order-hold, parameterization of the input. Denoting $\mathbf{x}_k = \mathbf{x}(t_k)$ and integrating the continuous dynamics in (11) over an interval leads to a discrete time representation of the dynamics

$$\mathbf{f}^d_k(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k + \int_{t_k}^{t_k+\delta t} \mathbf{f}^c(\mathbf{x}(\tau), \mathbf{u}_k, t)d\tau. \tag{26}$$

The integral in (26) is numerically approximated with an integration method of choice to achieve the desired approximation accuracy of the evolution of the continuous-time system under the zero-order-hold commands. We use an explicit second-order Runge–Kutta scheme.

The general nonlinear MPC problem presented below can be formulated by defining and evaluating a cost function and constraints on the grid of nodes

$$\min_{\mathbf{X},\,\mathbf{U}} \quad \Phi(\mathbf{x}_N) + \sum_{k=0}^{N-1} l_k(\mathbf{x}_k, \mathbf{u}_k) \tag{27a}$$

$$\text{s.t.} \quad \mathbf{x}_0 - \hat{\mathbf{x}} = \mathbf{0}, \tag{27b}$$

$$\mathbf{x}_{k+1} - \mathbf{f}^d_k(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}, \qquad k = 0, \ldots, N-1 \tag{27c}$$

$$\mathbf{g}_k(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}, \qquad k = 0, \ldots, N-1 \tag{27d}$$

where $\mathbf{X} = [\mathbf{x}_0^\top, \ldots \mathbf{x}_N^\top]^\top$ and $\mathbf{U} = [\mathbf{u}_0^\top, \ldots \mathbf{u}_{N-1}^\top]^\top$ are the sequences of state and input variables, respectively. The nonlinear cost and constraint functions l_k and $\mathbf{g}_k$ are discrete samples of the continuous counterpart. Collecting all decision variables into a vector, $\mathbf{w} = [\mathbf{X}^\top, \mathbf{U}^\top]^\top$, problem (27) can be written as a general NLP

$$\min_{\mathbf{w}} \quad \phi(\mathbf{w}), \quad \text{s.t.} \quad \begin{bmatrix} \mathbf{F}(\mathbf{w}) \\ \mathbf{G}(\mathbf{w}) \end{bmatrix} = \mathbf{0} \tag{28}$$

where $\phi(\mathbf{w})$ is the cost function, $\mathbf{F}(\mathbf{w})$ is the collection of initial state and dynamics constraints, and $\mathbf{G}(\mathbf{w})$ is the collection of all general equality constraints.

B. Sequential Quadratic Programming

SQP-based methods apply Newton-type iterations to Karush–Kuhn–Tucker optimality conditions, assuming some regularity conditions on the constraints [87]. The Lagrangian of the NLP in (28) is defined as

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}_\mathbf{G}, \boldsymbol{\lambda}_\mathbf{H}) = \phi(\mathbf{w}) + \boldsymbol{\lambda}_\mathbf{F}^\top \mathbf{F}(\mathbf{w}) + \boldsymbol{\lambda}_\mathbf{G}^\top \mathbf{G}(\mathbf{w}) \tag{29}$$

with Lagrange multipliers $\boldsymbol{\lambda}_\mathbf{F}$ and $\boldsymbol{\lambda}_\mathbf{G}$, corresponding to the dynamics and equality constraints. The Newton iterations can be equivalently computed by solving the following potentially nonconvex QP [88]:

$$\min_{\delta\mathbf{w}_i} \ \nabla_\mathbf{w}\phi(\mathbf{w}_i)^\top \delta\mathbf{w}_i + \frac{1}{2}\delta\mathbf{w}_i^\top \mathbf{B}_i \delta\mathbf{w}_i \tag{30a}$$

$$\text{s.t} \ \ \mathbf{F}(\mathbf{w}_i) + \nabla_\mathbf{w}\mathbf{F}(\mathbf{w}_i)^\top \delta\mathbf{w}_i = \mathbf{0} \tag{30b}$$

$$\mathbf{G}(\mathbf{w}_i) + \nabla_\mathbf{w}\mathbf{G}(\mathbf{w}_i)^\top \delta\mathbf{w}_i = \mathbf{0} \tag{30c}$$

where the decision variables, $\delta\mathbf{w}_i$, define the update step relative to the current iteration $\mathbf{w}_i$, and the Hessian $\mathbf{B}_i = \nabla^2_\mathbf{w}\mathcal{L}(\mathbf{w}_i, \boldsymbol{\lambda}_\mathbf{F}, \boldsymbol{\lambda}_\mathbf{G})$. Computing the solution to (30) provides a candidate decision variable update, $\delta\mathbf{w}_i$, and updated Lagrange multipliers.

C. Quadratic Approximation Strategy

As we seek to deploy MPC on dynamic robotic platforms, it is critical that the optimization problem in (30) is well conditioned and does not provide difficulty to numerical solvers. In particular, when $\mathbf{B}_i$ in (30a) is p.s.d, the resulting QP is convex and can be efficiently solved [68].

To ensure this, an approximate p.s.d Hessian is used instead of the full Hessian of the Lagrangian. For the tracking costs (16), the objective function has a least-squares form, in which case the generalized Gauss–Newton approximation

$$\nabla^2_\mathbf{w}\left(\frac{1}{2}\|\boldsymbol{\epsilon}_i(\mathbf{w})\|^2_{\mathbf{W}_i}\right) \approx \nabla_\mathbf{w}\boldsymbol{\epsilon}_i(\mathbf{w})^\top \mathbf{W}_i \nabla_\mathbf{w}\boldsymbol{\epsilon}_i(\mathbf{w}) \tag{31}$$

proves effective in practice [89]. Similarly, for the soft constraints, we exploit the convexity of the penalty function applied to the nonlinear constraint [90]

$$\nabla^2_\mathbf{w}\left(\mathcal{B}(\mathbf{h}(\mathbf{w}))\right) \approx \nabla_\mathbf{w}\mathbf{h}(\mathbf{w})^\top \nabla^2_\mathbf{h}\mathcal{B}(\mathbf{h}(\mathbf{w}))\nabla_\mathbf{w}\mathbf{h}(\mathbf{w}) \tag{32}$$

where the diagonal matrix $\nabla^2_\mathbf{h}\mathcal{B}(\mathbf{h}(\mathbf{w}))$ maintains the curvature information of the convex penalty functions. The contribution of the constraints to the Lagrangian in (29) is ignored in the approximate Hessian since we do not have additional structure that allows a convex approximation.

D. Constraint Projection

The equality constraints in Section IV-G were carefully chosen to have full row rank w.r.t. the control inputs, such that, after linearization, $\nabla_\mathbf{w}\mathbf{G}(\mathbf{w}_i)^\top$ has full row rank in (30c). This means that the equality constraints can be eliminated before solving the QP through a change of variables [88]

$$\delta\mathbf{w}_i = \mathbf{P}\delta\tilde{\mathbf{w}}_i + \mathbf{p} \tag{33}$$

where the linear transformation satisfies

$$\nabla_\mathbf{w}\mathbf{G}(\mathbf{w}_i)^\top \mathbf{P} = \mathbf{0}, \quad \nabla_\mathbf{w}\mathbf{G}(\mathbf{w}_i)^\top \mathbf{p} = -\mathbf{G}(\mathbf{w}_i). \tag{34}$$

After substituting (33) into (30), the following QP is solved w.r.t. $\delta\tilde{\mathbf{w}}_i$

$$\min_{\delta\tilde{\mathbf{w}}_i} \ \nabla_{\tilde{\mathbf{w}}}\tilde{\phi}(\mathbf{w}_i)^\top \delta\tilde{\mathbf{w}}_i + \frac{1}{2}\delta\tilde{\mathbf{w}}_i^\top \tilde{\mathbf{B}}_i \delta\tilde{\mathbf{w}}_i \tag{35a}$$

$$\text{s.t} \ \ \tilde{\mathbf{F}}(\mathbf{w}_i) + \nabla_{\tilde{\mathbf{w}}}\tilde{\mathbf{F}}(\mathbf{w}_i)^\top \delta\tilde{\mathbf{w}}_i = \mathbf{0}. \tag{35b}$$

Because each constraint applies only to the variables at one node k, the coordinate transformation maintains the sparsity pattern of an optimal control problem and can be computed in parallel. Since this projected problem now only contains costs and system dynamics, solving the QP only requires one Riccati-based iteration [71]. The full update $\delta\mathbf{w}_i$ is then obtained through back substitution into (33).

E. Line Search

To select an appropriate step size, we employ a line search based on the filter line search used in IPOPT [62]. In contrast to a line search based on a merit function, where cost and constraints are combined to one metric, the main idea is to ensure that each update either improves the constraint satisfaction or the cost function. The constraint satisfaction $\theta(\mathbf{w})$ is measured by taking the norm of all constraints scaled by the time discretization

$$\theta(\mathbf{w}) = \delta t \left|\left|\left[\mathbf{F}(\mathbf{w})^\top, \mathbf{G}(\mathbf{w})^\top\right]^\top\right|\right|_2. \tag{36}$$

In case of high or low constraint satisfaction, the behavior is adapted: When the constraint is violated beyond a set threshold, $\theta_{\max}$, the focus changes purely to decreasing the constraints; when constraint violation is below a minimum threshold, $\theta_{\min}$, the focus changes to minimizing costs.

Compared to the algorithm presented in [62], we remove recovery strategies and second-order correction steps, for which there is no time in the online setting. Furthermore, the history of iterates plays no role since we perform only one iteration per problem.

The simplified line search as used in this article is given in Algorithm 2 and contains three distinct branches in which a step can be accepted. The behavior at high constraint violation is given by line 9, where a step is rejected if the new constraint violation is above the threshold and worse than the current violation. The switch to the low constraint behavior is made in line 13: if both new and old constraint violations are low and the current step is in a descent direction, we require that the cost decrease satisfies the Armijo condition in line 14. Finally, the primary acceptance condition is given in line 18, where either a cost or constraint decrease is requested. The small constants γ_ϕ and γ_θ are used to fine-tune this condition with a required nonzero decrease in either quantity.

VI. Motion Execution

The optimized motion planned by the MPC layer consists of contact forces and desired joint velocities. We linearly interpolate the MPC motion plan at the 400-Hz execution rate and apply the feedback gains derived from the Riccati backward pass to

Algorithm 2: Backtracking Line Search.

```
1:  **Hyperparameters:** $\alpha_{\min} = 10^{-4}, \theta_{\max} = 10^{-2}, \theta_{\min} = 10^{-6}, \eta = 10^{-4}, \gamma_\phi = 10^{-6}, \gamma_\theta = 10^{-6}, \gamma_\alpha = 0.5$
2:  $\alpha \leftarrow 1.0$
3:  $\theta_k \leftarrow \theta(\mathbf{w}_i)$
4:  $\phi_k \leftarrow \phi(\mathbf{w}_i)$
5:  Accepted ← False
6:  while Not Accepted and $\alpha \geq \alpha_{\min}$ do
7:    $\theta_{i+1} \leftarrow \theta(\mathbf{w}_i + \alpha\delta\mathbf{w}_i)$
8:    $\phi_{i+1} \leftarrow \phi(\mathbf{w}_i + \alpha\delta\mathbf{w}_i)$
9:    if $\theta_{i+1} > \theta_{\max}$ then
10:     if $\theta_{i+1} < (1-\gamma_\theta)\theta_i$ then
11:       Accepted ← True
12:     end if
13:   else if $\max(\theta_{i+1}, \theta_i) < \theta_{\min}$ and $\nabla\phi(\mathbf{w}_i)^\top\delta\mathbf{w}_i < 0$
      then
14:     if $\phi_{i+1} < \phi_i + \eta\alpha\nabla\phi(\mathbf{w}_i)^\top\delta\mathbf{w}_i$ then
15:       Accepted ← True
16:     end if
17:   else
18:     if $\phi_{i+1} < \phi_i - \gamma_\phi\theta_i$ or $\theta_{i+1} < (1-\gamma_\theta)\theta_i$ then
19:       Accepted ← True
20:     end if
21:   end if
22:   if Not Accepted then
23:     $\alpha \leftarrow \gamma_\alpha\alpha$
24:   end if
25: end while
26: if Accepted then
27:   $\mathbf{w}_{i+1} \leftarrow \mathbf{w}_i + \alpha\delta\mathbf{w}_i$
28: else
29:   $\mathbf{w}_{i+1} \leftarrow \mathbf{w}_i$
30: end if
```

the measured state [45], [85]. The corresponding torso acceleration is obtained through (8). The numerical derivative of the planned joint velocities is used to determine a feedforward joint acceleration. A high-frequency whole-body controller (WBC) is used to convert the desired acceleration tasks into torque commands [91], [92], [93]. A generalized momentum observer is used to estimate the contact state [94]. In addition, the estimated external torques are filtered and added to the MPC and WBC dynamics, as described in [46]. We use the same filter setup as shown in [46, Fig. 13].

A. Event-Based Execution

Inevitably, the measured contact state will be different from the planned contact state used during the MPC optimization. In this case, the designed contact forces cannot be provided by the WBC. We have implemented simple reactive behaviors to respond to this situation and provide feedback to the MPC layer.

In case there is a planned contact, but no contact is measured, we follow a downward *regaining* motion for that foot. Under the assumption that the contact mismatch will be short, the MPC will start a new plan again from a closed contact state. In addition, we propagate the augmented system in (9) with the information that no contact force was generated, i.e., $\mathbf{0} \overset{!}{=} \mathbf{C}_\lambda\mathbf{s}_\lambda + \mathbf{D}_\lambda\boldsymbol{\nu}_\lambda$. In this way, the MPC layer will generate contact forces that maintain the requested smoothness w.r.t. the executed contact forces.

When contact is measured, but no contact was planned, the behavior depends on the planned time till contact. If contact was planned to happen soon, the measured contact is sent to the MPC to generate the next plan from that early contact state. In the meantime, the WBC maintains a minimum contact force for that foot. If no upcoming contact was planned, the measured contact is ignored.

TABLE II
WHOLE-BODY CONTROL TASKS

Type	Task
$=$	Floating base equations of motion. No motion at the contact points.
$\geq$	Torque limits. Friction cone constraint. Joint limit barrier constraint.
$w_i^2\|\cdot\|^2$	Swing leg motion tracking ($w_i = 100.0$). Torso linear and angular acceleration ($w_i = 1.0$). Contact force tracking. ($w_i = 0.01$).

B. Whole-Body Control

The WBC approach considers the full nonlinear rigid-body dynamics of the system in (6), including the estimate of disturbance forces. Each task is formulated as an equality constraint, inequality constraint, or least-squares objective affine in the generalized accelerations, torques, and contact forces. While we have used a hierarchical resolution of tasks in the past [93], in this article, we instead use a single QP and trade off the tracking tasks with weights. We found that a strict hierarchy results in a dramatic loss of performance in lower priority tasks when inequality constraints are active. In addition, the complexity of solving multiple QPs and null-space projections in the hierarchical approach is no longer justified with the high-quality motion reference coming from the MPC.

The complete list of tasks is given in Table II. The first two blocks of tasks enforce physical consistency and inequality constraints on torques, forces, and joint configurations. The joint limit constraint is derived from an exponential control barrier function (CBF) [95] on the joint limits, $\underline{\mathbf{q}}_j \leq \mathbf{q}_j \leq \overline{\mathbf{q}}_j$, resulting in the following joint acceleration constraints:

$$\ddot{\mathbf{q}}_j + (\gamma_1 + \gamma_2)\dot{\mathbf{q}}_j + \gamma_1\gamma_2(\mathbf{q}_j - \underline{\mathbf{q}}_j) \geq \mathbf{0} \tag{37}$$

$$-\ddot{\mathbf{q}}_j - (\gamma_1 + \gamma_2)\dot{\mathbf{q}}_j + \gamma_1\gamma_2(\overline{\mathbf{q}}_j - \mathbf{q}_j) \geq \mathbf{0} \tag{38}$$

with scalar parameters $\gamma_1 > 0, \gamma_2 > 0$. Provided that the full QP remains feasible, these CBF constraints guarantee that the state constraints are satisfied for all time and under the full nonlinear dynamics of the system [96].

For the least-squares tasks, we track swing leg motion with higher weight than the torso reference. This prevents that the robot exploits the leg inertia to track torso references in underactuated directions, and it ensures that the foot motion is prioritized over torso tracking when close to kinematics limits. Tracking the contact forces references with a low weight regulates the force distribution in case the contact configuration allows for internal forces.

Finally, the torque derived from the WBC, $\boldsymbol{\tau}_{\text{wbc}} \in \mathbb{R}^{12}$, is computed. To compensate for model uncertainty for swing legs, the integral of joint acceleration error with gain $K > 0$ is added to the torque applied to the system

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_{i,\text{wbc}} - K \int_{t_0^{\text{sw}}}^{t} (\ddot{\mathbf{q}}_i - \ddot{\mathbf{q}}_{i,\text{wbc}})\, dt \tag{39}$$

$$= \boldsymbol{\tau}_{i,\text{wbc}} - K \left(\dot{\mathbf{q}}_i - \dot{\mathbf{q}}_i(t_0^{\text{sw}}) - \int_{t_0^{\text{sw}}}^{t} \ddot{\mathbf{q}}_{i,\text{wbc}} dt \right) \tag{40}$$

where t_0^{sw} is the start time of the swing phase. The acceleration integral can be implemented based on the measured velocity $\dot{\mathbf{q}}_i$ and the velocity at the start of the swing phase, $\dot{\mathbf{q}}_i(t^{\text{sw}})$, as shown in (40). Furthermore, the feedback term is saturated to prevent integrator windup. For stance legs, a proportional–derivative term is added around the planned joint configuration and contact consistent joint velocity.

VII. Results

ANYmal is equipped with either two dome-shaped RoboSense bpearl LiDARs, mounted in the front and back of the torso, or with four Intel RealSense D435 depth cameras mounted on each side of the robot. Elevation mapping runs at 20 Hz on an onboard GPU (Jetson AGX Xavier). Control and state estimation are executed on the main onboard CPU (Intel i7-8850H, 2.6 GHz, Hexa-core) at 400 Hz, asynchronously to the MPC optimization, which is triggered at 100 Hz. Four cores are used for parallel computation in the MPC optimization. A time horizon of $T = 1.0$ s is used with a nominal time discretization of $\delta t \approx 0.015$ s, with a slight variation due to the adaptive discretization around gait transitions. Each multiple-shooting MPC problem, therefore, contains around 5000 decision variables. Part (a) and (b) of perception pipeline in Fig. 3 are executed on a second onboard CPU of the same kind and provides the precomputed layers over Ethernet.

To study the performance of the proposed controller, we report results in different scenarios and varying levels of detail. All perception, MPC, and WBC parameters remain constant throughout the experiments and are the same for simulation and hardware. An initial guess for these parameters was found in simulation, and we further fine-tuned them on hardware. First, results for the perception pipeline in isolation are presented in Section VII-A. Second, we validate the major design choices in simulation in Section VII-B. Afterward, the proposed controller is put to the test in challenging simulation, as well as hardware experiments in Section VII-C. All experiments are shown in the supplemental video [1]. Finally, known limitations are discussed in Section VII-D.

A. Perception Pipeline

The output of the steppability classification and plane segmentation (part (a) in Fig. 3) for a demo terrain is shown in Fig. 6. This terrain is available as part of the grid map library and contains a collection of slopes, steps, curvatures, rough terrain, and missing data. The left middle image shows that slopes and steps are, in general, well segmented. In the bottom right image, one sees the effect of the plane segmentation on a curved surface. In those cases, the terrain will be segmented into a collection of smaller planes. Finally, the rough terrain sections shown in the right middle and bottom left image show that the method is able to recognize such terrain as one big planar section as long as the roughness is within the specified tolerance. These cases also show the importance of allowing holes in the segmented regions, making it possible to exclude just those small regions where the local slope or roughness is outside the tolerance. A global convex decomposition of the map would result in many more regions.

The computation time for the construction and querying of the signed distance field is benchmarked on submaps of varying sizes extracted from the demo map (see Fig. 7). As expected,

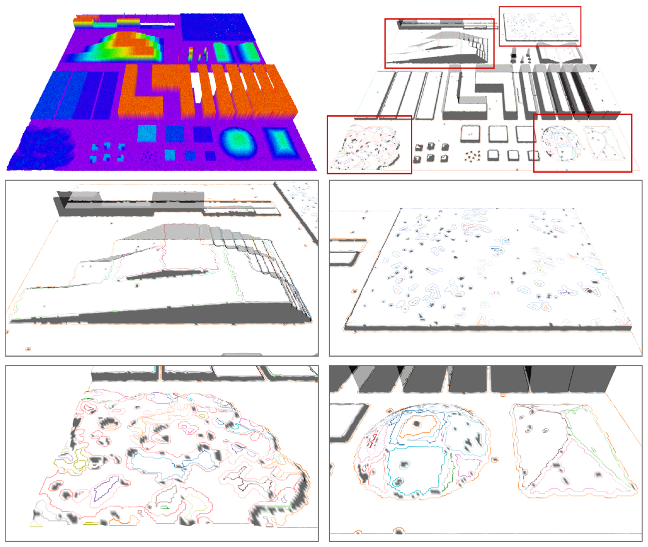


Fig. 6. Evaluation of the plane segmentation on a demo terrain [73]. The shown map has a true size of $20 \times 20 \times 1$ m with a resolution of 4 cm. Top left shows the elevation map with additive uniform noise of ± 2 cm plus Gaussian noise with a standard deviation of 2 cm. Top right shows the map after inpainting, filtering, steppability classification, and plane segmentation. Below, four areas of interest are shown. Their original location in the map is marked in the top right image.

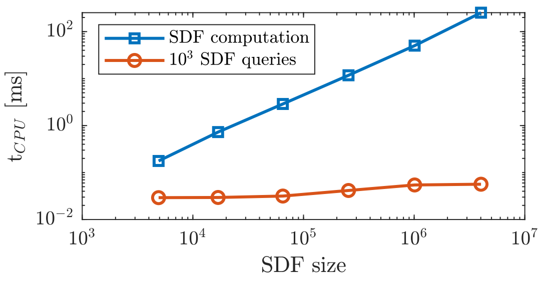


Fig. 7. Computation time for constructing and querying the signed distance field. Submaps of the terrain in Fig. 6 are used. *SDF size* on the horizontal axis denotes the total amount of data points in the SDF (width $\times$ length$\times$height). The query time is reported for the total of 10^3 random queries for the interpolated value and derivative.

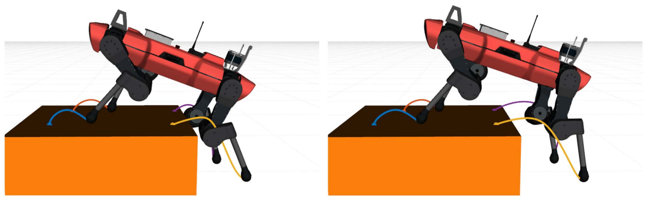


Fig. 8. ANYmal stepping up a box of 35 cm. Left: Without considering knee collisions. Right: Knee collision included in the optimization.

the construction time scales linearly with the SDF size, and the query time is constant with a slight increase when the memory size exceeds a cache level. During runtime, the local SDF size is typically below 10^5 voxels, resulting in a computation time well below 10 ms. Together with the map update rate of 20 Hz, the proposed method provides the SDF at an order of magnitude faster than methods that maintain a general 3-D voxel grid, with update rated reported around 1 Hz [79]. Per MPC iteration, around 10^3 SDF queries are made, making the SDF query time negligible compared to the total duration of one MPC iteration.

B. Simulation

1) Collision Avoidance: To highlight the importance of considering knee collisions with the terrain, the robot is commanded to traverse a box of 35 cm with a trotting gait at 0.25 ms^{-1}. Fig. 8 compares the simulation result of this scenario with and without the knee collisions considered. The inclusion of knee collision avoidance is required to successfully step up the box with the hind legs. As shown in the figure, the swing trajectories are altered. Furthermore, the base pose and last stepping location before stepping up are adjusted to prepare for the future, showing the benefit of considering all degrees of freedom in one optimization. Similarly, on the way down, the foothold optimization (within constraints) allows that the feet are placed away from the step, avoiding knee collisions while stepping down.

Fig. 9 provides insight into the solver during the motion performed with the knee collisions included. The four peaks in the cost function show the effect of the collision avoidance penalty when the legs are close to the obstacle during the step up and step down. Most of the time, the step obtained from the QP subproblem is accepted by the line search with the full step size of 1.0. However, between 7 and 8 s, the step size is decreased to prevent the constraint violation from further rising. This happens when the front legs step down the box and are close to collision. In those cases, the collision avoidance penalty is highly nonlinear, and the line search is required to maintain the right balance between cost decrease and constraint satisfaction. We note that the line search condition for low constraint violation is typically not achieved when using only one iteration per MPC problem.

2) Model Selection: In the same scenario, we compare the performance of the proposed dynamics for the base with those of the commonly used -SRBD. To be precise, the torso dynamics in (8) are evaluated at a constant nominal joint configuration and with zero joint velocities, while the rest of the controller remains

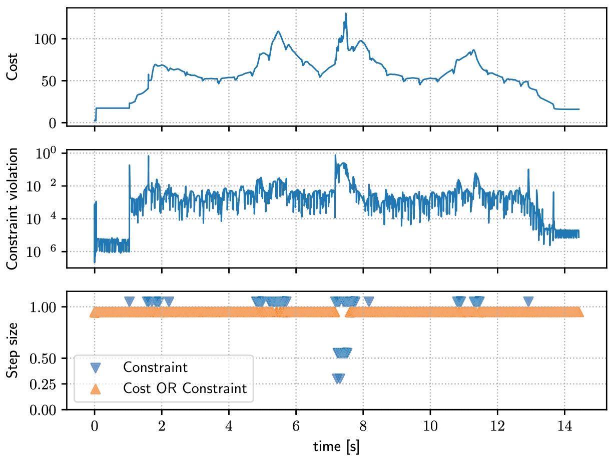


Fig. 9. Solver status during the box traversal motion (including knee collision avoidance). The first and second plots show the total cost, and constraint violation according to (36), after each iteration. The bottom plot shows the step size and the line search branch that led to the step acceptance. “Constraint” refers to a step accepted in the high constraint violation branch in line 9 of Algorithm 2, “Cost OR Constraint” refers to the branch where either cost or constraint decrease is accepted in line 18. Note that the low constraint violation branch, line 13, did not occur in this experiment.

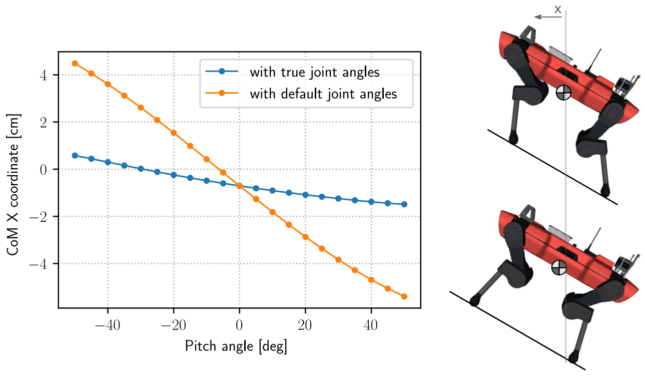


Fig. 10. Location of the center of mass (CoM) in heading direction for various torso pitch angles. The first set of CoM locations is evaluated with the *true* joint angles, which are obtained when aligning the legs with the gravity direction as in the top image. This corresponds to the reference in Section IV-E, which is tracked by the MPC. The second set of CoM locations is evaluated for the *default* joint angles, shown in the bottom image, as assumed by the SRBD model.

identical. When using the SRBD, the model does not describe the backward shift in the center of mass location caused by the leg configuration. The result is that the controller with the SRBD model has a persisting bias that makes the robot almost tip over during the step up. This model error is quantified in Fig. 10. At 30° pitch angle, there is a center of mass error of 2.6 cm, resulting in a bias of 13.3 N · m at the base frame. For reference, this is equivalent to an unmodeled payload of 3.6 kg at the tip of the robot. The proposed model fully describes the change in inertia and center of mass location and, therefore, does not have any issue to predict the state trajectory during the step up motion.

3) Solver Comparison: To motivate our choice to implement a multiple-shooting solver and move away from the DDP-based methods used in previous work, we compare both approaches on flat terrain and the stepping stone scenario shown in Fig. 11.

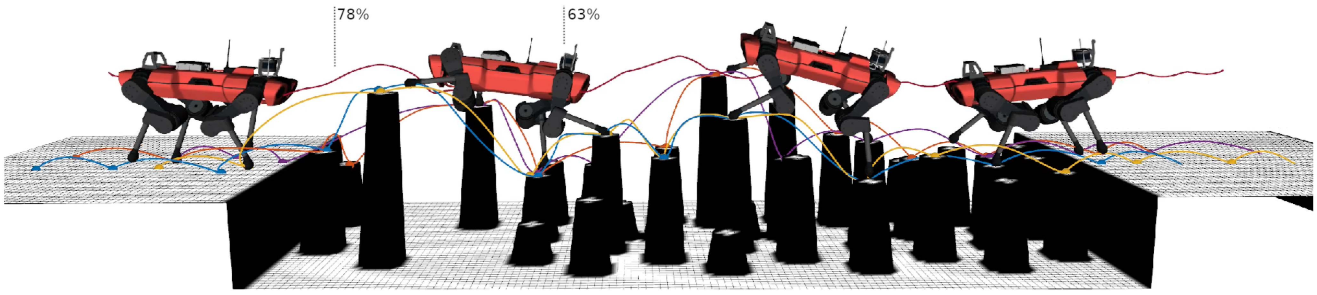


Fig. 11. ANYmal traversing stepping stones in simulation (right to left). The resulting state trajectories for feet and torso, and the snapshots are shown for a traversal with the multiple-shooting solver and a trotting gait at 0.75 ms^{-1}. The marked 63% and 78% locations indicate where the alternative solver, iLQR, diverges for 0.5 and 0.75 ms^{-1}, respectively.

TABLE III
SOLVER COMPARISON ON FLAT TERRAIN AND STEPPING STONES

	Baseline	Multiple shooting	iLQR
Flat, 0.50 m/s			
Cost	54.19	54.17	54.18
Dynamics constraints	1.70×10^{-7}	3.41×10^{-3}	0.0
Equality constraints	2.28×10^{-6}	3.51×10^{-3}	3.51×10^{-3}
Stones, 0.25 m/s			
Cost	151.92	156.22	156.69
Dynamics constraints	3.58×10^{-5}	1.01×10^{-2}	0.0
Equality constraints	7.24×10^{-4}	2.28×10^{-2}	2.14×10^{-2}
Stones, 0.50 m/s			
Cost	155.06	165.72	diverged at 78% scenario progress
Dynamics constraints	2.18×10^{-5}	1.53×10^{-2}	
Equality constraints	3.93×10^{-4}	3.82×10^{-2}	
Stones, 0.75 m/s			
Cost	199.49	215.98	diverged at 63% scenario progress
Dynamics constraints	1.20×10^{-4}	2.36×10^{-2}	
Equality constraints	1.29×10^{-3}	5.61×10^{-2}	

In particular, we compare against iLQR [64] and implement it with the same constraint projection, line search, and the Riccati Backward pass of HPIPM, as described in Section V. The key difference between the algorithms lies in the update step. For multiple shooting, we update both state and inputs directly: $\mathbf{u}_k^+ = \mathbf{u}_k + \alpha\delta\mathbf{u}_k$, $\mathbf{x}_k^+ = \mathbf{x}_k + \alpha\delta\mathbf{x}_k$. In contrast, iLQR proceeds with a line search over closed-loop nonlinear *rollouts* of the dynamics

$$\mathbf{u}_k^+ = \mathbf{u}_k + \alpha\mathbf{k}_k + \mathbf{K}_k\left(\mathbf{x}_k^+ - \mathbf{x}_k\right) \tag{41}$$

$$\mathbf{x}_{k+1}^+ = \mathbf{f}_k^d(\mathbf{x}_k^+, \mathbf{u}_k^+), \qquad \mathbf{x}_0^+ = \hat{\mathbf{x}} \tag{42}$$

where $\mathbf{K}_k$ is the optimal feedback gain obtained from the Riccati Backward pass and $\mathbf{k}_k = \delta\mathbf{u}_k - \mathbf{K}_k\delta\mathbf{x}_k$ is the control update. Due to this inherently single-threaded process, each line search for iLQR takes four times as long as for the multithreaded multiple shooting. However, note that with the hybrid multiple-shooting iLQR variants in [70], this difference vanishes.

Table III reports the solvers' average cost, dynamics constraint violation, and equality constraint violation for a trotting gait in several scenarios. As a baseline, we run the multiple-shooting solver until convergence (with a maximum of 50 iterations) instead of real-time iteration. To test the MPC in isolation, we use the MPC dynamics as the simulator and apply the MPC input directly. Because of the nonlinear rollouts of iLQR, dynamics constraints are always satisfied, and iLQR, therefore, has the edge over multiple shooting on this metric. However, as the scenario gets more complex and the optimization problem becomes harder, there is a point where the forward rollout of iLQR is unstable and diverges. For the scenario shown in Fig. 11, this happens in the place where the robot is forced to take a big leap at the 63% mark and at the 78% mark where the hind leg is close to singularity as the robot steps down. The continuous-time-variant SLQ [39] fails in similar ways. These failure cases are sudden, unpredictable, and happen regularly when testing on hardware, where imperfect elevation maps, real dynamics, and disturbances add to the challenge. The absence of long horizon rollouts in the multiple-shooting approach makes it more robust and better suited for the scenarios shown in this article. For cases where both solvers are stable, we find that the small dynamics violation left with multiple shooting in a real-time iteration setting does not translate to any practical performance difference on hardware. Finally, even for the most challenging scenario, multiple shooting with real-time iteration remains within 10% cost of the baseline.

However, we did find a fixed value for μ that provides satisfying performance for all scenarios. A too small value neglects constraint satisfaction, while a too large value induces small step sizes that destabilize the real-time iteration scheme. It is well known that the penalty parameter needs to be updated across iterations [88]. However, doing so in the context of real-time iteration is still an active area of research [97].

4) Contact Feedback: The reactive behavior under a mismatch in planned and sensed contact information is shown in the accompanying video. First, the sensed terrain is set to be 10 cm above the actual terrain, causing a late touchdown. Afterward, the sensed terrain is set 5 cm below the actual terrain, causing an early touchdown. The resulting vertical foot velocity for both cases is overlaid and plotted in Fig. 12. For the case of a late touchdown, the reactive downward accelerating trajectory is triggered as soon as it is sensed that contact is absent. For the early touchdown case, there is a short delay in detecting that contact has happened, but once contact is detected, the measured contact is included in the MPC, and the new trajectory is immediately replanned from the sensed contact location.

5) Stairs: The generality of the approach with respect to the gait pattern is demonstrated in the accompanying video by

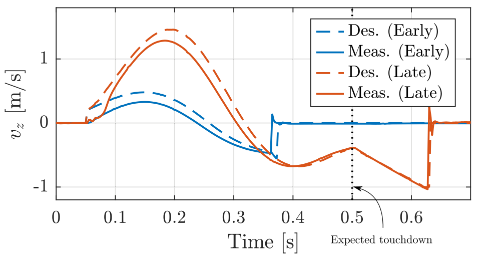


Fig. 12. Desired and measured vertical foot velocity for the early and late touchdown scenarios shown in the accompanying video. The vertical line at 0.5 s indicated the planned touchdown time.

executing a trot at 0.25 ms^{-1}, a pace at 0.3 ms^{-1}, a dynamic walk at 0.25 ms^{-1}, and a static walk at 0.2 ms^{-1} on a stairs with 18.5 cm rise and 24 cm run. Depending on the particular gait pattern and commanded velocity, the method autonomously decides to progress, repeat, or skip a step. Note that there are no parameters or control modes specific to the gait or the stair climbing scenario. All motions emerge automatically from the optimization of the formulated costs and constraints.

6) Obstacle Course: The controller is given a constant forward velocity command on a series of slopes, gaps, stepping stones, and other rough terrains. We traverse the terrain with a pace at 0.4 ms^{-1}, and a fast trotting gait with flight phase at 0.8 ms^{-1}. Fig. 13 shows the obstacle course and snapshots of the traversal with the fast trot. The supplemental video shows the planned trajectories for the feet together with the convex foothold constraints. On the right-hand side of the screen, a front view is shown together with the elevation map and plane segmentation below. The slower gaits used in the previous section are able to complete the scenario as well, but their video is excluded as they take long to reach the end.

Finally, a transverse gallop gait is demonstrated on a series of gaps. Due to the torque limitations of the system and friction limits up the slope, this gait is not feasible on the more complex obstacle course.

7) Comparison Against RL: We compare our method against a perceptive RL-based controller [14] in the same obstacle course. We adapt the gait pattern of our controller to match the nominal gait used by the learned controller. The video shows that the learning-based controller can cross the unstructured terrain at the beginning and end of the obstacle course. However, it fails to use the perceptive information fully and falls between the stepping stones when starting from the left and off the narrow passage when starting from the right. While the RL controller was not specifically trained on stepping stones, this experiment highlights that current RL-based locomotion results in primarily reactive policies and struggles with precise coordination and planning over longer horizons. In contrast, using a model and online optimization along a horizon makes our proposed method generalize naturally to these more challenging terrains.

C. Hardware

1) Obstacle Course: The obstacle course simulation experiment is recreated on hardware in two separate experiments. First, we tested a sequence of a ramp, gap, and high step, as shown in Fig. 14. During the middle section of this experiment, the robot faces all challenges simultaneously: While the front legs are stepping up to the final platform, the hind legs are still dealing with the ramp and gap. In a second scenario, the robot is walking on a set of uneven stepping stones, as shown in Fig. 15. The main challenge here is that the planes on the stepping stones are small and do not leave much room for the MPC to optimize the footholds. We found that in this scenario, the inclusion of the kinematics and reactive foothold offset during the plane selection, as described in Section IV-E, are important. A remaining challenge here is that our plane segmentation does not consider consistency over time. In some cases, the small foothold regions on top of stepping stones might appear and disappear as feasible candidates. The supplemental video shows how in this case the planned foot trajectory can fail, and the reactive contact regaining is required to save the robot.

TABLE IV
COMPUTATION TIMES PER MAP UPDATE AND MPC ITERATION

	Mean [ms]	Max [ms]
Classification and Segmentation	38.8	76.6
Signed distance field	1.3	7.6
LQ approximation	3.6	6.2
QP solve	2.7	4.4
Line search	0.3	0.9
MPC iteration	6.6	9.8

Computation times are reported in Table IV. Per map update, most time is spent on terrain classification and plane segmentation. More specifically, the RANSAC refinement takes the most time and can cause a high worst-case computation due to its sampling-based nature. On average, the perception pipeline is able to keep up with the 20-Hz map updates.

For the MPC computation time, the “LQ approximation” contains the parallel computation of the linear–quadratic model and equality constraint projection (Algorithm 1, lines 2–4). “QP solve” contains the solution of the QP and the back substitution of the solution (Algorithm 1, lines 5 and 6). Despite the parallelization across four cores, evaluating the model takes the majority of the time, with the single core solving of the QP in second place. On average, the total computation time is sufficient for the desired update rate of 100 Hz. The worst-case computation times are rare, and we hypothesize that they are mainly caused by variance in the scheduling of the numerous parallel processes on the robot. For the line search, the relatively high maximum computation time is attained when several steps are rejected, and the costs and constraints need to be recomputed.

2) Stairs: We validate the stair climbing capabilities on two-step indoor stairs and on outdoor stairs. Fig. 16 shows the robot on its way down the outdoor stairs. For these experiments, we obtain the elevation map from [98]. With its learning-based approach, it provides a high-quality estimate of the structure underneath the robot. Note that this module only replaces the source of the elevation map in Fig. 3 and does not change the rest of our perception pipeline. Figs. 17 and 18 show the measured joint velocities and torques alongside the same quantities within the MPC solution for five strides of the robot walking up the stairs. The optimized MPC values are within the specified limits and close to the measured values.

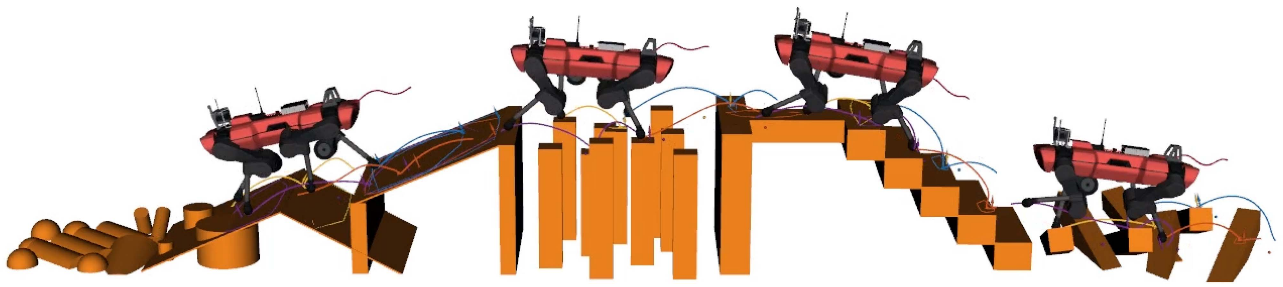


Fig. 13. ANYmal traversing an obstacle course in simulation (left to right). Snapshots are shown for a traversal with a trotting gait at 0.8 ms^{-1}. The MPC predictions are shown for each foot and for the torso center. For all contact phases within the horizon, the convex foot placement constraints are visualized.

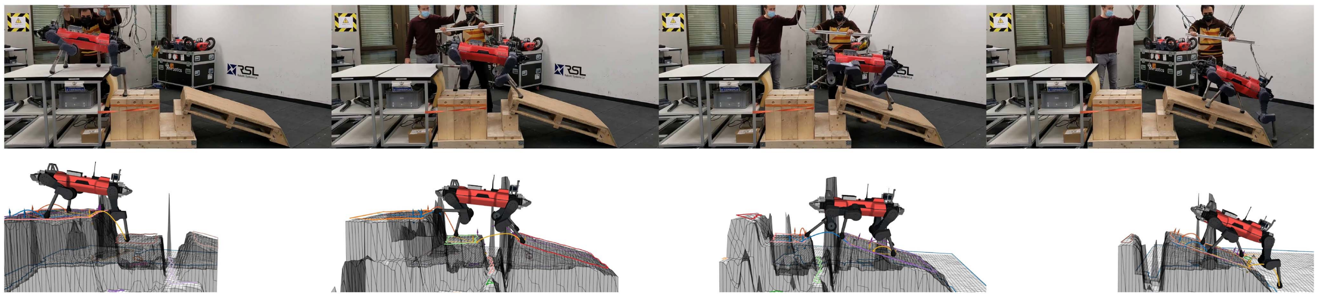


Fig. 14. Hardware experiment where ANYmal traverses a ramp, gap, and large step (from right to left). The bottom row shows the filtered elevation map, the foot trajectories over the MPC horizon, and the convex foothold constraints.

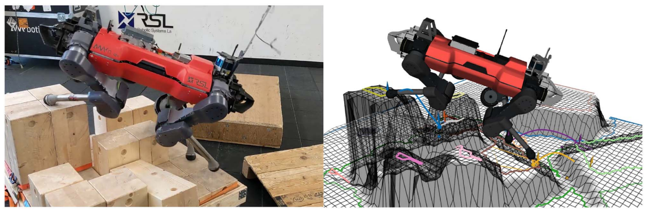


Fig. 15. Hardware experiment where ANYmal walks on top of uneven stepping stones. Each wooden block has an area of 20 × 20 cm^2 and each level of stepping stones is 20 cm higher than the previous one. The right image shows the filtered elevation map, the foot trajectories over the MPC horizon, and the convex foothold constraints.

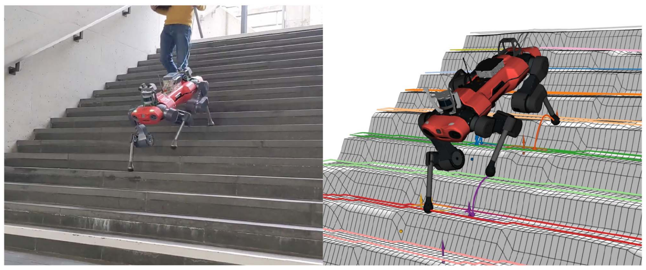


Fig. 16. Hardware experiment where ANYmal walks up and down outdoor stairs with a 16 cm rise and 29.5 cm run. The right image shows the filtered elevation map, the foot trajectories over the MPC horizon, and the convex foothold constraints.

D. Limitations

A fundamental limitation in the proposed controller is that the gait pattern is externally given and only adapted during early and late touchdown. Strong adverse disturbances, for example, in the direction of a foot that will soon lift, can make the controller fail. A change in the stepping pattern could be a much better response in such cases. Together with the reactive behaviors during contact mismatch, which are currently hardcoded, we see the potential for RL-based methods as a tracking controller to add to the robustness during execution.

Closely related to that, the current selection of the segmented plane and, therefore, the resulting foothold constraints happens independently for each leg. In some cases, this can lead to problems that could have been avoided if all legs were considered simultaneously. For example, while walking up the stairs sideways, all feet can end up on the same tread, leading to fragile support and potential self-collisions. Similarly, the presented method targets local motion planning and control, and we should not expect global navigation behavior. The current approach will attempt to climb over gaps and obstacles if so commanded by the user and will not autonomously navigate around them.

As with all gradient-based methods for nonlinear optimization, local optima and infeasibility can be an issue. With the simplification of the terrain to convex foothold constraints and by using a heuristic reference motion in the cost function, we have aimed to minimize such problems. Still, we find that in the case of very thin and tall obstacles, the optimization can get stuck. Fig. 19 shows an example where the foothold constraints lie behind the obstacle, and the reference trajectory correctly

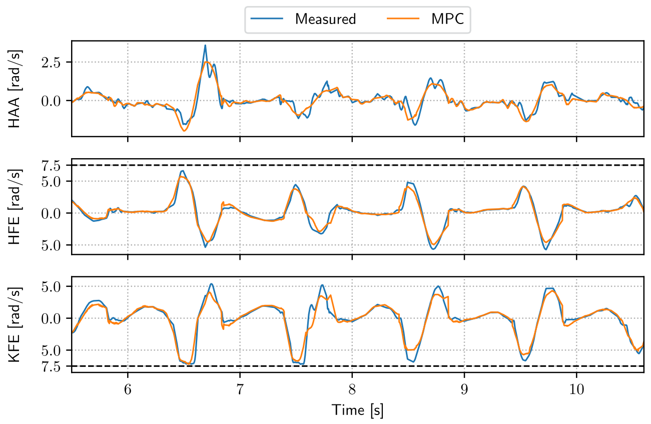


Fig. 17. Measured and MPC commanded joint velocities for the left front leg while walking up the stairs shown in Fig. 16. All joints, hip abduction aduction (HAA), hip flexion extension (HFE), and knee flexion extension (KFE), have a velocity limit of ± 7.5 rad s^{-1}.

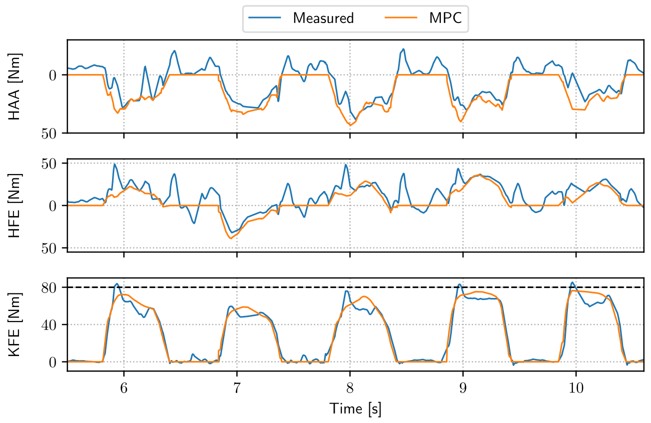


Fig. 18. Measured torque and approximated torque within the MPC formulation ($\boldsymbol{\tau}_i = \mathbf{J}_{j,i}^\top \boldsymbol{\lambda}_i$) for the left front leg while walking up the stairs shown in Fig 16. All joints, hip abduction aduction (HAA), hip flexion extension (HFE), and knee flexion extension (KFE), have a torque limit of ± 80 N $\cdot$ m.

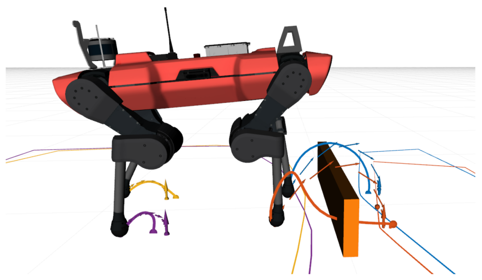


Fig. 19. Example of the MPC optimization being stuck inside a tall and thin structure of 5-cm width and 20-cm height. The feet reference trajectories used as part of the cost function are visualized as a sequence of arrows.

clears the obstacle. Unfortunately, one of the feet in the MPC trajectory goes right through the obstacle. Because all SDF gradients are horizontal at that part of the obstacle, there is no strong local hint that the obstacle can be avoided. For future work, we can imagine detecting such a case and triggering a sampling-based recovery strategy to provide a new collision-free initial guess. Alternatively, recent learning-based initialization could be employed [99], [100].

Finally, we show a gallop and trot with flight phases at the end of the video. For these motions, the perceptive information is turned OFF, and the robot estimates the ground plane through a history of contact points. It demonstrates that the presented MPC is ready to express and stabilize these highly dynamic motions. Unfortunately, the elevation map is not usable due to artifacts from impacts and state estimation drift.

VIII. Conclusion

In this article, we proposed a controller capable of perceptive and dynamic locomotion in challenging terrain. By formulating perceptive foot placement constraints through a convex inner approximation of steppable terrain, we obtain a nonlinear MPC problem that can be solved reliably and efficiently with the presented numerical strategy. Steppability classification, plane segmentation, and an SDF are all precomputed and updated at 20 Hz. Asynchronously precomputing this information minimizes the time required for each MPC iteration and makes the approach real-time capable. Furthermore, by including the complete joint configuration in the system model, the method can simultaneously optimize foot placement, knee collision avoidance, and underactuated system dynamics. With this rich set of information encoded in the optimization, the approach discovers complex motions autonomously and generalizes across various gaits and terrains that require precise foot placement and whole-body coordination.

Appendix A
Signed Distance Field Computation

This section details how a signed distance field can be computed for a 2.5-D elevation map. Consider the following general definition for the squared Euclidean distance between a point in space and the closest obstacle:

$$\mathcal{D}(x,y,z) = \min_{x',y',z'} \Big[(x-x')^2 + (y-y')^2 + (z-z')^2 + I(x',y',z') \Big] \tag{43}$$

where $I(x',y',z')$ is an indicator function returning 0 for an obstacle and ∞ for empty cells.

As described in [101], a full 3-D distance transform can be computed by consecutive distance transforms in each dimension of the grid, in arbitrary order. For the elevation map, the distance along the z-direction is trivial. Therefore, starting the algorithm with the z-direction simplifies the computation. First, (43) can be rewritten as follows:

$$\mathcal{D}(x,y,z) = \min_{x',y'} \Big[(x-x')^2 + (y-y')^2 + \min_{z'} \big[(z-z')^2 + I(x',y',z') \big] \Big] \tag{44}$$

$$= \min_{x',y'} \Big[(x-x')^2 + (y-y')^2 + f_z(x',y',z) \Big] \tag{45}$$

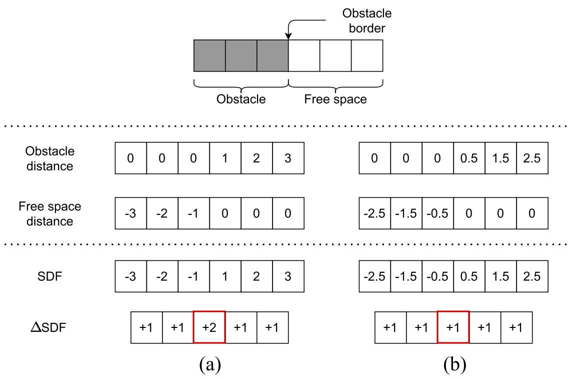


Fig. 20. 1-D example illustrating the effect of distance metric on the SDF. When taking the Euclidean distance between cell centers in (a), the SDF will have a discontinuous gradient across the obstacle border. Taking the distance between cell center and the border of an occupied/free cell as in (b) avoids this issue.

where $f_z(x', y', z)$ is a function that returns for each horizontal position, the 1-D distance transform in z-direction. For an elevation map, this function has the following closed-form solution at a given height z:

$$f_z(x', y', z) = \begin{cases} (z - h(x', y'))^2, & \text{if } z \geq h(x', y') \\ 0, & \text{otherwise} \end{cases} \tag{46}$$

where $h(x', y')$ denotes the evaluation of the elevation map.

The same idea can be used to compute the distance to obstacle-free space and obtain the negative valued part of the SDF. Adding both distances together provides the full SDF and gradients are computed by finite differences between layers, columns, and rows. However, naively taking the Euclidean distance between cell centers as the minimization of (45) leads to incorrect values around obstacle borders, as illustrated in Fig. 20. We need to account for the fact that the obstacle border is located between cells, not at the cell locations themselves. This can be resolved by adapting (45) to account for the discrete nature of the problem

$$\mathcal{D}(x, y, z) = \min_{\{x', y'\} \in \mathcal{M}} \left[d(x, x') + d(y, y') + f_z(x', y', z) \right] \tag{47}$$

where $\{x', y'\} \in \mathcal{M}$ now explicitly shows that we only minimize over the discrete cells contained in the map, and $d(\cdot, \cdot)$ is a function that returns the squared distance between the center of one cell and the border of another

$$d(x, x') = \begin{cases} (|x - x'| - 0.5r)^2, & \text{if } x \neq x' \\ 0, & \text{otherwise} \end{cases} \tag{48}$$

where r is the resolution of the map. The distance transforms can now be computed based on (47), for each height in parallel, with the 2-D version of the algorithm described in [101].

References

[1] Supplementary video. Accessed: May 23, 2023. [Online]. Available: https://youtu.be/v6MhPl2ICsc

[2] M. Tranzatto et al., "CERBERUS: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the DARPA subterranean challenge," *Field Robot.*, vol. 2, pp. 274–324, 2021.

[3] A. Bouman et al., "Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2518–2525.

[4] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 2665–2670.

[5] D. Belter, P. Labecki, and P. Skrzypczynski, "Adaptive motion planning for autonomous rough terrain traversal with a walking robot," *J. Field Robot.*, vol. 33, no. 3, pp. 337–370, 2016.

[6] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control," *IEEE Trans. Robot.*, vol. 36, no. 6, pp. 1635–1648, Dec. 2020.

[7] P. Fankhauser, M. Bjelonic, D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *Proc. Int. Conf. Robot. Autom.*, 2018, pp. 5761–5768.

[8] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," in *Proc. Int. Conf. Humanoid Robots (Humanoids)*, 2019, pp. 9–16.

[9] 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.

[10] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 4102–4109.

[11] 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.

[12] 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.

[13] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," in *Proc. Robot., Sci. Syst.*, 2021. [Online]. Available: https://www.roboticsproceedings.org/rss17/p061.html

[14] 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.

[15] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5370–5376, Oct. 2020.

[16] D. Kim et al., "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *Proc. Int. Conf. Robot. Autom.*, 2020, pp. 2464–2470.

[17] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "MPC-based controller with terrain insight for dynamic legged locomotion," in *Proc. Int. Conf. Robot. Autom.*, 2020, pp. 2436–2442.

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

[19] 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.

[20] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 295–302.

[21] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations," in *Proc. Int. Conf. Robot. Autom.*, 2020, pp. 1410–1416.

[22] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proc. Vol.*, vol. 17, no. 2, pp. 1603–1608, 1984.

[23] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, vol. 2. Madison, WI, USA: Nob Hill Publishing, 2017.

[24] F. Farshidian et al., "OCS2: An open source library for optimal control of switched systems." Accessed: May 23, 2023. [Online]. Available: https://github.com/leggedrobotics/ocs2

[25] M. Hutter et al., "ANYmal—A highly mobile and dynamic quadrupedal robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 38–44.

[26] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *Proc. Int. Conf. Robot. Autom.*, 2008, pp. 811–818.

[27] S. Tonneau, A. D. Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 586–601, Jun. 2018.

[28] M. Zucker et al., "CHOMP: Covariant hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9/10, pp. 1164–1193, 2013.

[29] C. Mastalli, I. Havoutis, A. W. Winkler, D. G. Caldwell, and C. Semini, "On-line and on-board planning and perception for quadrupedal locomotion," in *Proc. IEEE Int. Conf. Technol. Practical Robot Appl.*, 2015, pp. 1–7.

[30] M. Bajracharya, J. Ma, M. Malchano, A. Perkins, A. A. Rizzi, and L. Matthies, "High fidelity day/night stereo mapping with vegetation and negative obstacle detection for vision-in-the-loop walking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 3663–3670.

[31] S. Bazeille et al., "Quadruped robot trotting over irregular terrain assisted by stereo-vision," *Intell. Service Robot.*, vol. 7, no. 2, pp. 67–77, 2014.

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

[33] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1184–1189.

[34] O. A. V. Magana et al., "Fast and continuous foothold adaptation for dynamic locomotion through CNNs," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2140–2147, Apr. 2019.

[35] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 2908–2927, Oct. 2022.

[36] W. Yu et al., "Visual-locomotion: Learning to walk on complex terrains with vision," in *Proc. 5th Annu. Conf. Robot Learn.*, 2022, pp. 1291–1302.

[37] M. Vukobratović and B. Borovac, "Zero-moment point—Thirty five years of its life," *Int. J. Humanoid Robot.*, vol. 1, no. 1, pp. 157–173, 2004.

[38] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Adv. Robot.*, vol. 24, no. 5/6, pp. 719–737, 2010.

[39] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 93–100.

[40] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Auton. Robots*, vol. 35, no. 2/3, pp. 161–176, 2013.

[41] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified MPC framework for whole-body dynamic locomotion and manipulation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4688–4695, Jul. 2021.

[42] A. Meduri, P. Shah, J. Viereck, M. Khadiv, I. Havoutis, and L. Righetti, "BiConMP: A nonlinear model predictive control framework for whole body motion planning," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 905–922, Apr. 2023.

[43] D. Pardo, M. Neunert, A. Winkler, R. Grandia, and J. Buchli, "Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion," in *Proc. Robot., Sci. Syst. Conf.*, 2017. [Online]. Available: https://www.roboticsproceedings.org/rss13/p42.html

[44] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2703–2710.

[45] C. Mastalli et al., "Agile maneuvers in legged robots: A predictive control approach," 2022, *arXiv:2203.07554*.

[46] 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.

[47] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 577–584.

[48] 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.

[49] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 69–81, 2014.

[50] B. Aceituno-Cabezas et al., "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2531–2538, Jul. 2018.

[51] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 31–38.

[52] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3316–3323, Oct. 2018.

[53] B. Ponton, A. Herzog, A. D. Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–7.

[54] F. Farshidian, M. Kamgarpour, D. Pardo, and J. Buchli, "Sequential linear quadratic optimal control for nonlinear switched systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1463–1469, 2017.

[55] T. Seyde, J. Carius, R. Grandia, F. Farshidian, and M. Hutter, "Locomotion planning through a hybrid Bayesian trajectory optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 5544–5550.

[56] M. Herbert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Proc. Int. Conf. Robot. Autom.*, 1989, pp. 997–1002.

[57] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 279–286.

[58] S. Bertrand, I. Lee, B. Mishra, D. Calvert, J. Pratt, and R. Griffin, "Detecting usable planar regions for legged robot locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 4736–4742.

[59] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 8352–8358.

[60] 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.

[61] 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.

[62] 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.

[63] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. Amsterdam, The Netherlands: Elsevier, 1970.

[64] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 4906–4913.

[65] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 7674–7679.

[66] C. Mastalli et al., "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 2536–2542.

[67] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[68] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, "Recent advances in quadratic programming algorithms for nonlinear model predictive control," *Vietnam J. Math.*, vol. 46, no. 4, pp. 863–882, 2018.

[69] M. Diehl, H. Bock, and J. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.

[70] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A family of iterative Gauss-Newton shooting methods for nonlinear optimal control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[71] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[72] R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function," *Math. Program.*, vol. 91, no. 2, pp. 239–269, 2002.

[73] P. Fankhauser and M. Hutter, "A universal grid map library: Implementation and use case for rough terrain navigation," in *Robot Operating System (ROS)—The Complete Reference*, A. Koubaa, Ed., vol. 1. New York, NY, USA: Springer, 2016, ch. 5.

[74] 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.

[75] A. Chilian and H. Hirschmüller, "Stereo camera based navigation of mobile robots on rough terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 4571–4576.

[76] K. Wu, E. Otoo, and K. Suzuki, "Optimizing two-pass connected-component labeling algorithms," *Pattern Anal. Appl.*, vol. 12, no. 2, pp. 117–135, 2009.

[77] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," *Comput. Graph. Forum*, vol. 26, no. 2, pp. 214–226, 2007.

[78] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Comput. Vis., Graph., Image Process.*, vol. 30, no. 1, pp. 32–46, 1985.

[79] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6177–6184, Oct. 2020.

[80] P.-B. Wieber, "Holonomy and nonholonomy in the dynamics of articulated motion," in *Fast Motions in Biomechanics and Robotics*. New York, NY, USA: Springer, 2006, pp. 411–425.

[81] 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.

[82] J. Hauser and A. Saccon, "A barrier function method for the optimization of trajectory functionals with constraints," in *Proc. IEEE 45th Conf. Decis. Control*, 2006, pp. 864–869.

[83] C. Feller and C. Ebenbauer, "A stabilizing iteration scheme for model predictive control based on relaxed barrier functions," *Automatica*, vol. 80, pp. 328–339, 2017.

[84] R. Deits and R. Tedrake, *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*. Cham, Switzerland:Springer, 2015, pp. 109–124.

[85] 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.

[86] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *J. Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[87] O. Mangasarian and S. Fromovitz, "The Fritz John necessary optimality conditions in the presence of equality and inequality constraints," *J. Math. Anal. Appl.*, vol. 17, no. 1, pp. 37–47, 1967.

[88] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[89] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[90] R. Verschueren, N. van Duijkeren, R. Quirynen, and M. Diehl, "Exploiting convexity in direct optimal control: A sequential convex quadratic programming method," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 1099–1104.

[91] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 2641–2648.

[92] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Souères, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Trans. Robot.*, vol. 29, no. 2, pp. 346–362, Apr. 2013.

[93] C. Dario Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 558–564.

[94] G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim, "Contact model fusion for event-based locomotion in unstructured terrains," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 4399–4406.

[95] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *Proc. Amer. Control Conf.*, 2016, pp. 322–328.

[96] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. IEEE 53rd Conf. Decis. Control*, 2014, pp. 6271–6278.

[97] S. Na, "Global convergence of online optimization for nonlinear model predictive control," in *Proc. Adv. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, J. W. Vaughan, Eds., 2021, vol. 34, pp. 12441–12453.

[98] 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.

[99] O. Melon, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 9805–9811.

[100] T. S. Lembono, C. Mastalli, P. Fernbach, N. Mansard, and S. Calinon, "Learning how to walk: Warm-starting optimal control solver with memory of motion," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 1357–1363.

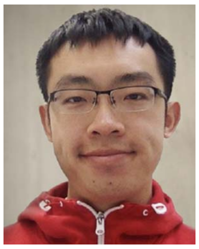
[101] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory Comput.*, vol. 8, no. 1, pp. 415–428, 2012.

Ruben Grandia received the B.Sc. degree in aerospace engineering from the Delft University of Technology, Delft, The Netherlands, in 2014, and the M.Sc. degree in robotics, systems, and control in 2017 from ETH Zurich, Zurich, Switzerland, where he is currently working toward the Ph.D. degree with the Robotic Systems Lab, under the supervision of Prof. M. Hutter.

His research interests include nonlinear optimal control and its application to dynamic mobile robots.

Fabian Jenelten received the B.Sc. and M.Sc. degrees in mechanical engineering in 2015 and 2018, respectively, from ETH Zurich, Zurich, Switzerland, where he is currently working toward the Ph.D. degree with the Robotic Systems Lab, under the supervision of Prof. M. Hutter.

His research interests include model- and learning-based control approaches for legged robot locomotion.



Shaohui Yang received the B.Eng. degree in computer science from Hong Kong University of Science and Technology, Hong Kong, in 2019, and the M.Sc. degree in systems, control, and robotics from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2022.

After conducting his master's thesis with the Robotic Systems Lab, ETH Zurich, he joined the Automatic Control Laboratory, Ècole polytechnique fèdèrale de Lausanne, Lausanne, Switzerland, as a Doctoral Assistant, under the supervision of Prof. C. Jones. His research interests include optimization, model-predictive control, and learning-based control.

Farbod Farshidian received the M.Sc. degree in electrical engineering from the University of Tehran, Tehran, Iran, in 2012, and the Ph.D. degree in planning and control of legged robots from ETH Zurich, Zurich, Switzerland, in 2017.

He is currently a Senior Scientist with Robotic System Lab, ETH Zurich. He is part of the National Centre of Competence in Research's Robotics and Digital Fabrication. His research interests include mobile robots' motion planning and control, aiming to develop algorithms and techniques to endow these platforms to operate autonomously in real-world applications.

Marco Hutter received the M.Sc. degree in mechanical engineering and the Ph.D. degree in design and control of legged robots with compliant actuation from ETH Zurich, Zurich, Switzerland, in 2009 and 2013, respectively.

He is currently an Associate Professor of Robotic Systems with ETH Zurich. He is part of the National Centre of Competence in Research's Robotics and Digital Fabrication and principal investigator in various international projects (e.g., EU NI) and challenges. His research interests include development of novel machines and actuation concepts together with the underlying control, planning, and machine learning algorithms for locomotion and manipulation.