Perceptive Locomotion in Rough Terrain – Online Foothold Optimization

Originally published in:
IEEE Robotics and Automation Letters 5(4), https://doi.org/10.1109/LRA.2020.3007427

Funding acknowledgement:
188596 - Perceptive Dynamic Locomotion on Rough Terrain (SNF)
780883 - subTerranean Haptic INvestiGator (EC)

Perceptive Locomotion in Rough Terrain – Online Foothold Optimization

Fabian Jenelten[1], Takahiro Miki[1], Aravind E Vijayan[2], Marko Bjelonic[1], Marco Hutter[1]

***Abstract*— Compared to wheeled vehicles, legged systems have a vast potential to traverse challenging terrain. To exploit the full potential, it is crucial to tightly integrate terrain perception for foothold planning. We present a hierarchical locomotion planner together with a foothold optimizer that finds locally optimal footholds within an elevation map. The map is generated in real-time from on-board depth sensors. We further propose a terrain-aware contact schedule to deal with actuator velocity limits. We validate the combined locomotion pipeline on our quadrupedal robot ANYmal with a variety of simulated and real-world experiments. We show that our method can cope with stairs and obstacles of heights up to *33 %* of the robot's leg length.**

I. Introduction

Currently, legged robots are majorly deployed on mildly rough ground, an environment where their wheeled counterparts outperform them in terms of energy efficiency and speed. However, when it comes to cluttered terrains, the adaptation of footsteps and foot clearance make legged locomotion potentially a highly successful concept.

Kolter et al. [1] presented one of the first terrain-aware control pipelines, which was applied to *LittleDog* using a statically stable gait. A linear combination of features extracted from height maps with different resolutions is used to generate a foot-cost map. Footholds are found by a greedy search. Kalakrishnan et al. [2] proposed several improvements to this approach. In particular, a foothold ranking function learned from expert demonstration is used for foothold selection. Both methods are based on accurately pre-scanned environments, pre-processed height maps, and motion capture systems. Relying on statically stable gaits, Fankhauser et al. [3] introduced a rough terrain locomotion planner where a foothold is selected from a binary foothold-score map. Results were presented with the robot *ANYmal*.

Mastalli et al. [4] presented an approach that simultaneously optimizes for center of mass (COM) pose and foothold locations, where footholds are subject to a terrain-cost map. Optimization duration is in the range of several minutes, hence requiring off-line pre-processing. Results have been shown on *HyQ* with a crawling gait. Using the same platform, Magaña et al. [5] introduced a self-supervised foothold classifier based on a convolutional neural network (CNN). The network learns a correction step based on a nominal foothold, a processed height map, and simplified kinematics of the robot. Other works [6] [7] have focused on footstep planners that lay out a complete pattern of footholds connecting the robot's current state with a target location. As these approaches require comparably large computation times, we are concentrating on local foothold adaptation only.

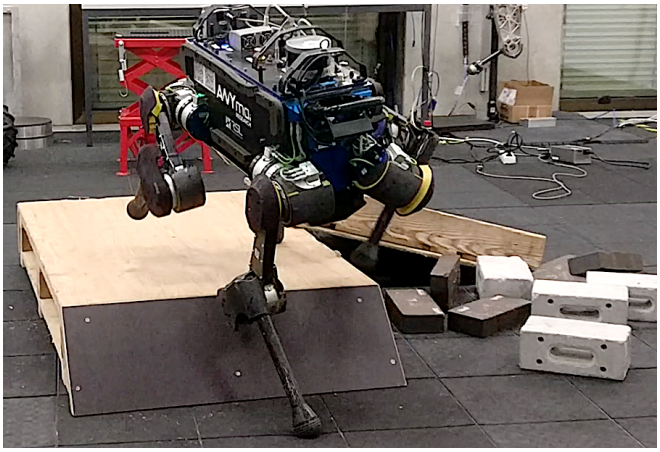

Fig. 1. ANYmal is equipped with a depth camera in the front of the torso to map the local terrain. A batch search optimization finds locally optimal footholds in an elevation map, allowing for agile locomotion in rough terrain.

The major contribution of this paper is a batch search approach that selects optimal footholds from a heightmap in the proximity of a nominal foothold. We compute a foothold-score similar to [3] and include the kinematics similar to [5]. The difference to the former is that we do not constrain locomotion to static gaits. Compared to the latter one, we achieve a sufficiently small computation time, rendering the training of a CNN obsolete. In the second part of this paper, we propose terrain-based gait adaption to overcome large obstacles. Finally, we demonstrate the success of our control architecture with ANYmal (Fig. 1), a rugged fully torque controllable quadruped robot.

II. Architectural Overview

The locomotion pipeline is following a hierarchical structure consisting of interacting modules, as illustrated in Fig 2. A key module is the batch search that, given a set of nominal footholds, computes a set of terrain aware footholds. On flat

This research was supported by the Swiss National Science Foundation (SNSF) as part of project No.188596.

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics (NCCR Robotics).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780883.

[1]Authors are with the Robotic Systems Lab, ETH Zurich, Switzerland. {fabian.jenelten, marko.bjelonic}@mavt.ethz.ch, {tamiki, mahutter}@ethz.ch

[2]Author is with ANYbotics AG, Oerlikon, Switzerland. avijayan@anybotics.com

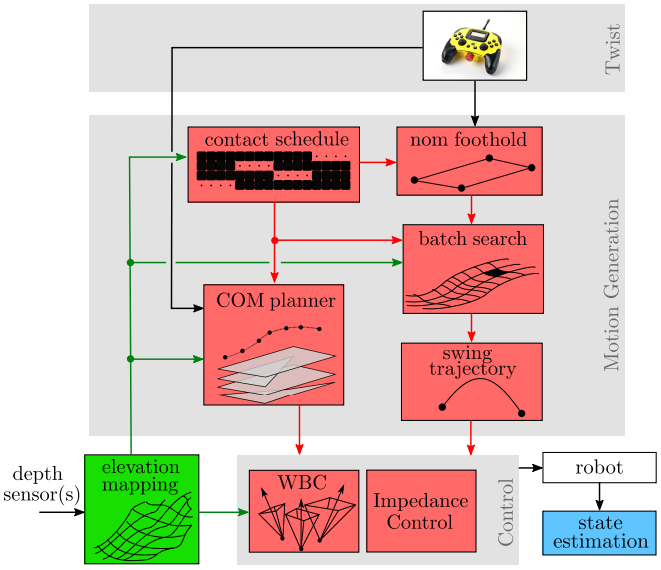

Fig. 2. Illustration of the perceptive locomotion pipeline. Blocks in red represent modules of the locomotion stack while blue and green blocks indicate purely sensor driven modules.

ground, the optimal foothold location is majorly affected by the contact timing and the desired velocity twist, while on rough ground, the gradient of the terrain pushes the footholds away from potential edges. The terrain is modeled as a discrete grid where each grid cell is associated with a height value (forming a so-called elevation map).

The lift-off sequence and the desired contact durations are determined by a contact schedule. A torso optimizer plans a 2d COM trajectory through the planned footholds, subject to a planar Zero-Moment Point (ZMP) stability criterion [8]. A swing leg trajectory is obtained by fitting conjoined splines through the desired foothold and the previous stance foot location [9].

At each time step, the desired torso and end-effector states are extracted from the associated trajectories. The reference signals are tracked through a hierarchical whole-body controller (WBC) [8], which takes into account the plane normals to ensure the contact forces remain within the local friction cones. An impedance control law runs in parallel to robustify locomotion close to singular configurations and to stiffen the motion in case of slippage.

III. Mapping

The terrain in the local vicinity of the robot is represented by an elevation map [10], which is obtained online by projecting measurement points from any depth camera(s) into a grid map [11].

A typical problem observed in mapping is the odometry drift, causing artifacts in the map and a position drift between the map and the robot. Fankhauser et al. [10] proposed a sensor fusion strategy to build a robot-centric map. We found, however, the fusion step computationally too expensive to be deployed on a highly dynamic platform. We also do not want to rely on external motion capture systems [2] since those are not available in real-world scenarios.

We implement a drift compensation filter for the height.[1] The mismatch between the robot state and the map is reduced by adjusting the current map height to be consistent with the latest sensor scan. Additionally, we integrate other filters used for edge sharpening, visibility cleanup, terrain normal computation, and in-painting (obstacle occlusion). A detailed presentation of the elevation mapping is out of the scope of this paper and will be discussed in our future work.

The drift compensation filter allows us to update mapping and state estimation without any correction step, e.g. map fusion. This also means that the forward kinematics is computed using only the robot's estimated state and does not account for any height mismatch relative to the map.

While walking down, the end-effectors of the front legs are likely to enter the field of view of the depth camera, leading to spikes in the elevation map. We tackle this problem by rejecting depth measurements located on the feet.

IV. Foothold Optimization

The task of finding a set of optimal footholds is split into N independent batch-search optimizations, where N is the number of legs. Since we only consider robots with point feet, the orientation of the feet is not taken into account.

A. Notations

For a quadrupedal robot, we use LF, RF, LH, RH to denote the left-front, right-front, left-hind, and right-hind leg. We define a foothold $\mathbf{p}_i$ of leg $i \in \{\text{LF, RF, LH, RH}\}$ by the location of a single grid cell and the associated height. An optimal foothold is indicated by writing $\mathbf{p}_i^*$.

If not stated differently, all vectors are given in the world frame. For some tasks, we use the *control frame* $\mathcal{C}$. The x-axis of the control frame is aligned with the robot's heading direction while roll and pitch follow from the *local terrain plane* $\mathcal{P}(\mathbf{n}_{\text{plane}})$, characterized by the terrain normal $\mathbf{n}_{\text{plane}}$ [12]. The terrain plane is obtained by fitting a plane through the current (if grounded) and previous (if swinging) stance foot positions $\{\mathbf{p}_{i,\text{stance}}\}_{i=1,\dots N}$.

In the following, $\hat{\mathbf{p}}_{i,\text{thigh}}$ will refer to the ith measured limb-thigh position defined as the rotation center of the hip-flexion joint. We further use the notation $\mathbf{p}_{i,\text{thigh,lift-off}}$ and $\mathbf{p}_{i,\text{thigh,touch-down}}$ for the predicted limb-thigh position at lift-off and touch-down time, respectively. As a rough estimate, it can be computed with a velocity projection $\hat{\mathbf{p}}_{i,\text{thigh}} + \mathbf{R}_{WC}\mathbf{v}^{\mathcal{C}}_{\text{des}} t_{i,\text{lift-off/touch-down}}$ with $\mathbf{v}^{\mathcal{C}}_{\text{des}}$ the desired linear velocity in control frame, $\mathbf{R}_{WC}$ the rotation matrix control to world frame and $t_{i,\text{lift-off/touch-down}}$ the time duration extracted from the contact schedule

$$t_{i,\text{lift-off}} = \begin{cases} t \text{ until swing} & i \text{ grounded} \\ t \text{ until stance + stance duration} & i \text{ swinging} \end{cases}$$
$$t_{i,\text{touch-down}} = t \text{ until stance.} \tag{1}$$

[1] We note that the odometry drift is significantly larger along z of the world frame compared to x and y. We thus only consider a height correction.

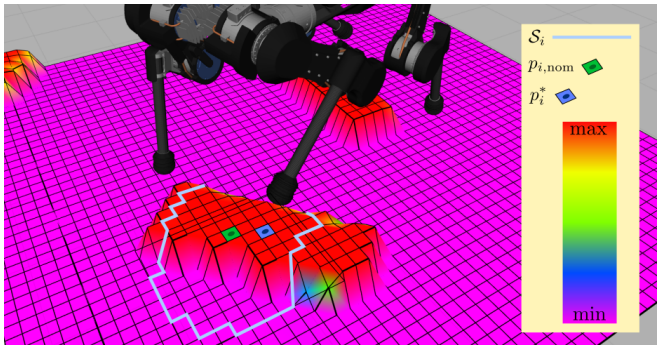

Fig. 3. Illustration of the batch-search approach for leg $i = \text{LF}$. Each cell of the grid map contains a value for the height (represented by different colors). We seek for an optimal foothold $\mathbf{p}_i^*$ within the search space $\mathcal{S}_i$ (or submap) centered about the nominal foothold $\mathbf{p}_{i,\text{nom}}$.

In our previous work [13], we introduced a *path regularizer* computed from a desired velocity twist. We use this high-level trajectory to extract the associated thigh positions.

B. Batch Search Approach

For each foot i, we iterate through the search area $\mathcal{S}_i$, a submap of the grid map (see Fig. 3). The grid-map has dimensions $1.6\,\text{m} \times 1.6\,\text{m}$ and consists of square cells with dimensions $0.035\,\text{m} \times 0.035\,\text{m}$, each holding a value for the height. The search space $\mathcal{S}_i$ is limited by a circle with radius $0.35\,\text{m}$ centered around a *nominal foothold* $\mathbf{p}_{i,\text{nom}}$. For each grid cell in the search area, we check for feasibility and compute an objective value. The optimal foothold $\mathbf{p}_i^*$ is found as the feasible grid cell with smallest cost.

Unlike [7], the search space is not restricted to convex patches and hence does not require surface segmentation. As opposed to similar approaches [3] [5], we use the nominal foothold only for the search-space selection but disregard it afterward. In fact, we want to avoid biasing the optimization towards a foothold location selected under blind conditions.

C. Objectives

For each grid cell, we compute an objective function value as the sum of weighted objectives. The various contributing objectives are detailed below, listed in descending order according to their weighting (importance).

1) Default Leg Configuration: A core task is to find footholds that realize a desired velocity twist while being able to react to external disturbances. We suggest to enforce a default foothold pattern by writing

$$\min_{\mathbf{p}_i \in \mathcal{S}_i} \; ||\mathbf{p}_{i,\text{thigh,lift-off}} + \mathbf{p}_{i,\text{thigh}\to\text{foot}} - \mathbf{p}_i||_2 + ||\mathbf{p}_{i,\text{thigh,touch-down}} + \mathbf{p}_{i,\text{thigh}\to\text{foot}} - \mathbf{p}_i||_2, \tag{2}$$

where $\mathbf{p}_{i,\text{thigh}\to\text{foot}}$ is a vector pointing from the limb thigh to a default foot location. For many gaits, the following choice results in satisfactory results

$$\mathbf{p}_{i,\text{thigh}\to\text{foot}} = -l_\text{nom} \cdot \left(\kappa \cdot \mathbf{n}_{z,\text{world}} + (1-\kappa) \cdot \mathbf{n}_\text{plane}\right), \tag{3}$$

with l_nom the nominal leg extension, $\mathbf{n}_{z,\text{world}}$ the z-axis of the world frame, $\mathbf{n}_\text{plane}$ the normal of the local terrain plane and $\kappa \in [0,1]$. Choosing $\kappa < 1$ will make the robot lean back when climbing up which helps to avoid knee joint collisions with obstacles. This problem is farther explained in Section VI-E.

For the sake of disturbance rejection we slightly modify the objective (2) by replacing $\mathbf{p}_{i,\text{thigh,lift-off}}$ with

$$\tilde{\mathbf{p}}_{i,\text{thigh,lift-off}} = \mathbf{p}_{i,\text{thigh,lift-off}} + k_v \cdot \mathbf{e}_i, \tag{4}$$

where $\mathbf{e}_i = \hat{\mathbf{v}}_{i,\text{thigh}} - \mathbf{v}_\text{des}$ is the thigh velocity error, $\hat{\mathbf{v}}_{i,\text{thigh}}$ the measured velocity of the limb thigh and $k_v > 0$ the corresponding gain for weighting the velocity feedback. The same applies also for the predicted limb touch-down location.

2) Foothold Score: The foothold score $s_f(h) \in [0,1]$ is an additional layer in the grid map computed from the heightmap h. A value close to 0 corresponds to flat ground which is considered a "safe" foot location while a value close to 1 indicates edges, slopes, or roughness.

Similar to [3], we compute the foothold score as a linear combination of quality measures, in our case

$$s_f = \underbrace{\lambda_1 \cdot \sigma(s(h))}_{\text{edges}} + \underbrace{\lambda_2 \cdot \bar{s}(h)^2}_{\text{slopes}} + \underbrace{\lambda_3 \cdot |h - \bar{h}|}_{\text{roughness}}, \tag{5}$$

with the weights $\lambda_1 + \lambda_2 + \lambda_3 = 1$, the slope $s(h)$ of the elevation (angle between gravity vector and normal), the standard deviation $\sigma(s(h))$, the mean $\bar{s}(h)$ of the slope, and the mean $\bar{h}$ of the height. All quantities are normalized to the unit interval using the maximum slope $\pi/2$ and the maximum allowed step height.

3) Push Over: Compared to [5], we update the footholds continuously even beyond the point where a swing leg reaches the height apex. This greatly improves reactive behavior. Unfortunately, jumps in the foothold location of swinging legs can cause aggressive end-effector motions. Those jumps can be particularly severe if the search space contains many edges. A possible solution is to minimize the maximum foothold-score between two consecutive foothold updates. Let $s_{f,\max}(\mathbf{p}_{i,\text{prev}}^*, \mathbf{p}_i)$ denote the maximum foothold-score on a line connecting the optimal previous foothold $\mathbf{p}_{i,\text{prev}}^*$ with $\mathbf{p}_i$, then:

$$\min_{\mathbf{p}_i \in \mathcal{S}_i} \begin{cases} s_{f,\max}(\mathbf{p}_{i,\text{prev}}^*, \mathbf{p}_i) & i \text{ swinging} \\ 0 & i \text{ grounded.} \end{cases} \tag{6}$$

4) Support Area: To increase the support area, we define the following objective function

$$\min_{\mathbf{p}_i \in \mathcal{S}_i} \; \left\{ \sum_{j \neq i} ||\hat{\mathbf{p}}_j - \mathbf{p}_i||_2 \right\}^{-1}. \tag{7}$$

The vector $\hat{\mathbf{p}}_j$ is the measured end-effector location of a neighboring leg j. Since we also enforce hard constraints IV-D.2, the objective is well defined for all feasible cases.

5) Previous Foothold: The collision avoidance constraint outlined in Section IV-D.2 can cause a foothold to jump from one to the other side of a neighboring end-effector within two consecutive updates. We address this issue by invoking

$$\min_{\mathbf{p}_i \in \mathcal{S}_i} \; ||\mathbf{p}_{i,\text{prev}}^* - \mathbf{p}_i||_2. \tag{8}$$

6) Leg Over-Extension: The following objective is used to slightly regularize the optimization towards footholds that minimize leg extension

$$\min_{\mathbf{p}_i \in \mathcal{S}_i} \; ||\mathbf{p}_{i,\text{thigh,lift-off}} - \mathbf{p}_i||_2 + ||\mathbf{p}_{i,\text{thigh,touch-down}} - \mathbf{p}_i||_2. \quad (9)$$

D. Constraints

Grid cells that violate one of the constraints presented below will be discarded.

1) Max Step Height: Taking high steps increases the possibility of operating the joints at large torques and close to singular configurations. Let $h_{\text{obstacle}}(\mathbf{p}_i, \mathbf{p}_{i,\text{stance}})$ be the largest obstacle height on a line between the foothold $\mathbf{p}_i$ and the current or previous stance foot location $\mathbf{p}_{i,\text{stance}}$. We impose a hard constraints on the step height by enforcing

$$\begin{aligned} h_{\text{obstacle}}(\mathbf{p}_i, \mathbf{p}_{i,\text{stance}}) - \mathbf{p}_{i,z} &< h_{\max} \\ h_{\text{obstacle}}(\mathbf{p}_i, \mathbf{p}_{i,\text{stance}}) - \mathbf{p}_{i,\text{stance},z} &< h_{\max}. \end{aligned} \quad (10)$$

2) Leg Collision: To avoid end-effector collision, which typically occur during side-stepping, we reject footholds close to neighboring end-effectors

$$||\hat{\mathbf{p}}_j - \mathbf{p}_i||_2 > d_{\min}, \qquad \forall i \neq j. \quad (11)$$

3) Leg Over-Extension: The prediction of the limb thigh IV-C.1 is not very accurate, and therefore, hard constraints on leg over-extension are typically too conservative. We accept the occurrence of kinematically unfeasible footholds and deal with over-extension by impedance control V-B.

E. Nominal Foothold

The computation of the nominal foothold, used to define the search region, is derived as the analytical solution of the objective (2)

$$\begin{aligned} \mathbf{p}_{i,\text{nom}} = \; & 0.5 \cdot (\tilde{\mathbf{p}}_{i,\text{thigh,lift-off}} + \tilde{\mathbf{p}}_{i,\text{thigh,touch-down}}) \\ & + \mathbf{p}_{i,\text{thigh}\to\text{foot}}. \end{aligned} \quad (12)$$

F. Optimization Times and Implementation Details

Each grid cell is checked for feasibility before calculating the objectives. By designing all objectives globally positive, we can skip a grid cell as soon as the local objective exceeds the minimum value found so far.

The batch search takes on average $1.8\,\text{ms}$ to find a set of four footholds. The maximum duration is approximate twice the average and is reached if all objectives have to be evaluated. Compared to the CNN framework [5], computation times are up to 12.5 times larger. The optimization duration is in the range of the control sampling time $2.5\,\text{ms}$, but could be further improved using a CNN.[2]

If the nominal foothold is not within the grid map, e.g., because the robot moved faster than the heightmap was updated, it is treated as an optimal foothold.

[2]We note that optimization times increase quadratically with the resolution of the grid.

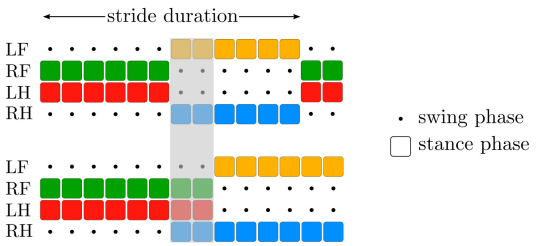

Fig. 4. The top image shows a trotting gait in time domain for one stride. The bottom image shows the perturbed gait after LF leg exhibits a large step. The swing time for LF is increased by delaying the touch-down event. Since phase events with three supporting legs are preferred over one, the lift-off events for RF and LF are delayed by the same amount of time.

V. Planning and Control

The locomotion controller is split across several modules from which one is the foothold optimizer. In the following, we will further highlight some modules introduced in Fig. 2 that have been adapted to the elevation map or that are of special importance for perceptive locomotion.

A. Contact Schedule

We use a contact schedule that produces lift-off and touch-down timings for a variety of periodic gaits. On rough terrain, a fixed periodic gait may not be the best choice. For example, joint velocities of a high stepping leg are much larger than those that take a nominal step. Following the observation that the stride duration increases with the step-height in human gaits [14], we adapt gait parameters to the local terrain.

Let $h_{\text{obstacle}}(\mathbf{p}_1, \mathbf{p}_2)$ be the highest point on a line connecting two position vectors $\mathbf{p}_1$ and $\mathbf{p}_2$. We define the height above the previous stance foot and the height above the desired foothold as

$$\begin{aligned} h_{i,\text{stance}} &= h_{\text{obstacle}}(\mathbf{p}_{i,\text{stance}}, \mathbf{p}_i^*) - \mathbf{p}_{i,z,\text{stance}} \\ h_i^* &= h_{\text{obstacle}}(\mathbf{p}_{i,\text{stance}}, \mathbf{p}_i^*) - \mathbf{p}_{i,z}^*. \end{aligned} \quad (13)$$

We use the swing-over step height $h_{i,\max} = h_{i,\text{stance}} + h_i^*$ to increase the swing phase $t_{i,\text{swing}}$ according to

$$\tilde{t}_{i,\text{swing}} = t_{i,\text{swing}} + k_o \cdot h_{i,\max}, \quad (14)$$

with $k_o > 0$ the obstacle gain. To preserve the lift-off order, stance durations are adjusted accordingly. The approach is further exemplified in Fig. 4.

B. Impedance Control

Given the desired torso pose, the desired end-effector position and velocity for leg i are transformed into joint space through inverse (differential) kinematics, yielding desired joint positions $\mathbf{q}_{i,\text{des}}$ and velocities $\dot{\mathbf{q}}_{i,\text{des}}$. For the inverse kinematics, we use an analytical approach that selects the solution that preserves the leg configuration. The pseudo-inverse of the contact Jacobian used in the inverse differential kinematics is based on the damped least-squares method. The joint space references are used in an impedance control law of the form

$$\boldsymbol{\tau}_{i,\text{des}} = \boldsymbol{\tau}_{i,\text{wbc}} + \mathbf{K}_p \cdot (\mathbf{q}_i - \mathbf{q}_{i,\text{des}}) + \mathbf{K}_d \cdot (\dot{\mathbf{q}}_i - \dot{\mathbf{q}}_{i,\text{des}}), \quad (15)$$

whereby the desired joint torques $\boldsymbol{\tau}_{i,\text{wbc}}$ are computed by the WBC i.e. inverse dynamics, and the impedance gain matrices $\mathbf{K}_p$,$\mathbf{K}_d$ are diagonal positive definite. Joint level impedance control is a particularly useful tool to handle leg over-extension as the inverse kinematics can avoid the singularity. It further allows to stiffen the joints in case of slip events, as discussed in our previous work [15].

C. Torso Orientation

The locomotion controller will align the torso with the control frame $\mathcal{C}$. To adapt torso orientation based on the elevation map, we fit the local terrain plane through current and desired (instead of previous stance) foot positions.

VI. Experiments

We have tested the perceptive locomotion pipeline on ANYmal [16], a fully torque controllable quadruped robot, designed for harsh in- and outdoor environments. For this work, the robot is equipped with an Intel RealSense d435 which publishes a point cloud at 6 Hz. Control and state estimation runs on an on-board PC (Intel i7-7600U, 3.5 Ghz, dual-core 64-bit) at a frequency of 400 Hz.

A. GPU Based Map Representation

Using a CPU implementation, we found that the updating rate of the map[3] is too slow to be paired with a fast-walking robot. We, therefore, developed a GPU-based grid map using CuPy [17], resulting in 14 times faster map updates.

We measured the on-board calculation time of our mapping pipeline. From the depth camera, we get around 61 624 points per cloud at 6 Hz. The whole calculation time (from receiving the point cloud to copying from GPU memory and publishing the processed map) is 28.9 ms on average while the processing time inside GPU is around 8.4 ms. In our case, elevation mapping runs on an on-board Jetson AGX Xavier which updates the map at the same frequency as the point cloud.

B. Stair Gap Climbing (Simulation)

Inspired by the experimental setup presented in [5], we build a simulated world consisting of 5 beams with depth 0.12 m equally distributed along the x-axis of the world frame. Each beam is shifted by 0.24 m and enlarged in height by 0.05 m relative to its precursor, forming a stair-gap scenario (see Fig. 5). The depth of the last beam is widened to form a supporting platform. To demonstrate reliability, we consider data collected from a total of 12 trials with varying heading velocities, climbing directions and initial positions, executed with a trotting gait. We report the overall success rate as 100 %.

In the first experiment outlined in Fig. 5, the stair-gaps are crossed in ascending climbing direction with two different commanded heading velocities $\{0.3\,\text{m/s}, 0.5\,\text{m/s}\}$. For the low-speed experiments (0.3 m/s), the nominal step length is shorter than the distance between two beams. Hence, for maintaining a constant average torso velocity, the robot is required to step in place on several beams. This implies that there is an optimal heading velocity for which each foot touches down on each bar exactly once. We let such terrain depending input modulation left for our future work. Experiments executed with blind trot[4] always failed to overcome the second beam.

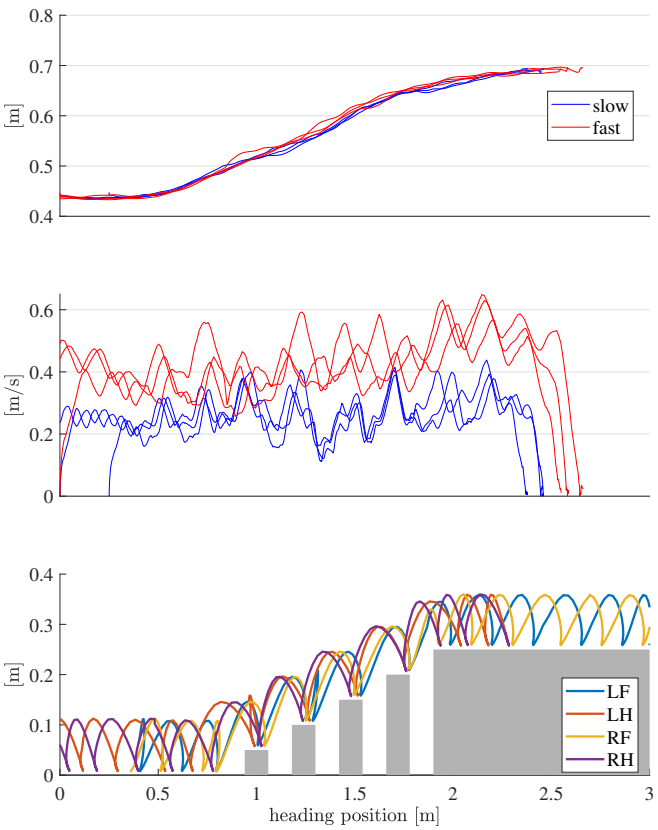

Fig. 5. Illustration of the bar-balancing experiment in planar xz plane for the ascending climbing direction. Top: measured torso height. Middle: measured torso horizontal velocity. Bottom: Measured end-effector trajectories for 0.3 m/s. The experiment was repeated 3 times each with 0.3 m/s and 0.5 m/s commanded heading velocity.

In Fig 6 we repeat the same experiments using the descending climbing direction. For the high-speed runs (0.5 m/s), the nominal step length is larger than the distance between two beams. Hence, the end-effectors do not step on each beam but rather over-swing some of them. Experiments simulated with blind trot always failed as the legs get stuck in between two beams.

C. Perceptive Locomotion over Rough Terrain

We demonstrate ANYmal's capability to traverse highly rough terrain. Fig. 7 contains some snapshots of the parkour run that was recorded over a period of 2 min continuous walking. The bricks were passed three, the slope two, and the platform four times.

[3]update rate on CPU (without map fusion) is ≈ 2.5 Hz.

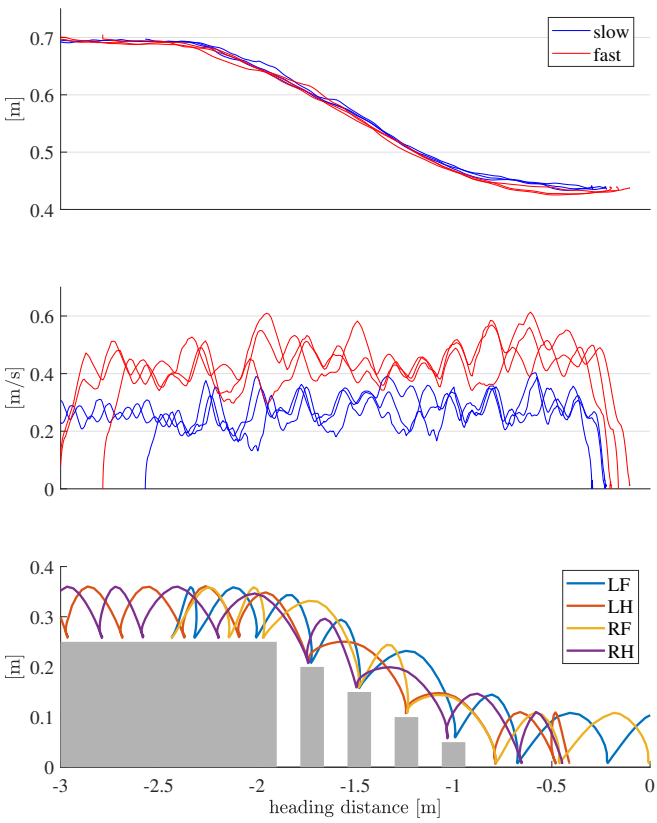

Fig. 6. Illustration of the bar-balancing experiment in planar xz plane for the descending climbing direction. Top: measured torso height. Middle: measured torso horizontal velocity. Bottom: Measured end-effector trajectories for $0.5\,\mathrm{m/s}$. The experiment was repeated 3 times each with $0.3\,\mathrm{m/s}$ and $0.5\,\mathrm{m/s}$ commanded heading velocity.

TABLE I

SUCCESS RATE FOR STAIR CLIMBING.

climbing direction	impedance control	success rate for stairs moderate	steep
up	enabled	18/18	10/18
	disabled	18/18	3/18
down	enabled	18/18	14/18
	disabled	18/18	12/18

D. Climbing Up and Down Stairs (Simulation)

In the simulation environment, we set up a two straight stairways (moderate and steep) consisting of 12 steps with depth $0.29\,\mathrm{m}$ and height $0.17\,\mathrm{m}$ (steep) or $0.10\,\mathrm{m}$ (moderate). An experiment starts with a varying heading position on the bottom/top floor and ends as soon as the robot falls or reaches the top/bottom floor. The commanded heading velocity is always $0.3\,\mathrm{m/s}$. We repeat each experiment 18 times for the steep and moderate staircase, for the ascending and descending climbing direction as well as with and without impedance control, yielding a total of 144 trials.

As table I implies, the success rate is higher for the descending direction when using the steep stairway. This observation can be explained by the absence of knee-joint collisions which mainly appear while climbing up. For the descending direction, failures were mostly due to leg over-extension of swinging legs. The results show that impedance control generally increases the success rate. Since it helps to avoid a leg entering the singular configuration, the evidence is particularly significant for the ascending climbing direction. In the case of stair climbing with moderate inclination, the robot never encounters knee joint collisions nor leg over-extension and consequently, the success rate is $100\,\%$.

[4]The blind trotting controller is equivalent to the perceptive controller, but excluding gait adaption and elevation map depending tasks.

E. Climbing Up Stairs

In a final experiment associated to Fig 8, we show dynamic stair climbing with stair parameters identical to the simulated "steep" experiment of section VI-D. Each experiment starts on a flat platform and ends if the robot falls. Across three experiments, failure was detected on average after 18 steps. Consistent with our simulations, failures were mostly due to knee-joint collisions.

VII. CONCLUSION AND FUTURE WORK

The presented foothold generation, which plans a single footstep ahead, has been shown to be applicable in various uneven environments. Swing phase adaptation was introduced as a useful concept to overcome large obstacles and impedance control was shown to robustify locomotion in the presence of knee-joint collisions. We validated the combined locomotion controller on ANYmal, demonstrating locomotion over high platforms, slopes, and movable bricks.

The stability criterion used in the COM planner is valid on flat ground only, and torso orientation and height are assumed to be constant over the prediction horizon. These are, in fact, very limiting assumptions. Moreover, the lack of a guarantee for kinematic feasibility together with potential knee joint collisions render especially stair climbing a tough problem. Nevertheless, we have shown that we can achieve reliable stair climbing by choosing stair parameters conservative enough to not provoke knee-joint collisions or leg over-extension. Experiments performed with stair parameters found in daily life highlight the potential of our method, but also reveal the disadvantage of omitting kinematic constraints. In order to generalize locomotion to even more challenging terrains, we plan to replace the ZMP-constraints of the COM planner with a dynamic stability criterion valid in rough environments. Motivated by our results, we also would like to address the problem of kinematic unfeasible footholds.

REFERENCES

[1] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 811–818.

[2] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 2665–2670.

[3] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5761–5768.

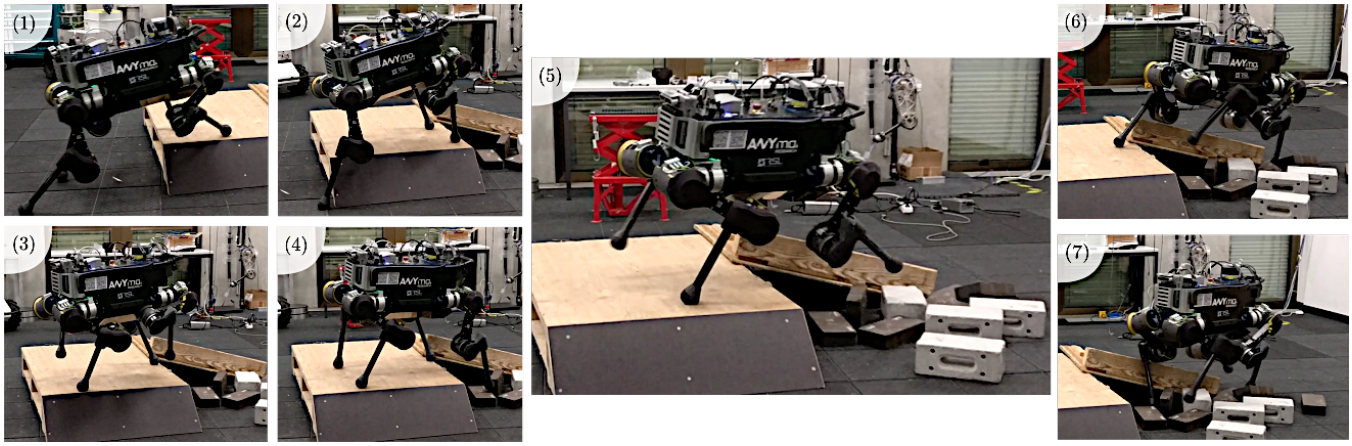

Fig. 7. The operator manually commands a heading velocity of $0.3\,\mathrm{m/s}$ and adjusts yaw velocity to navigate the robot into the most difficult parts of the obstacle parkour. The height of the platform is $0.22\,\mathrm{m}$ which corresponds to $33\,\%$ of the robot's maximum leg length. The slope and the bricks are not fixed and eventually move when stepping on it.

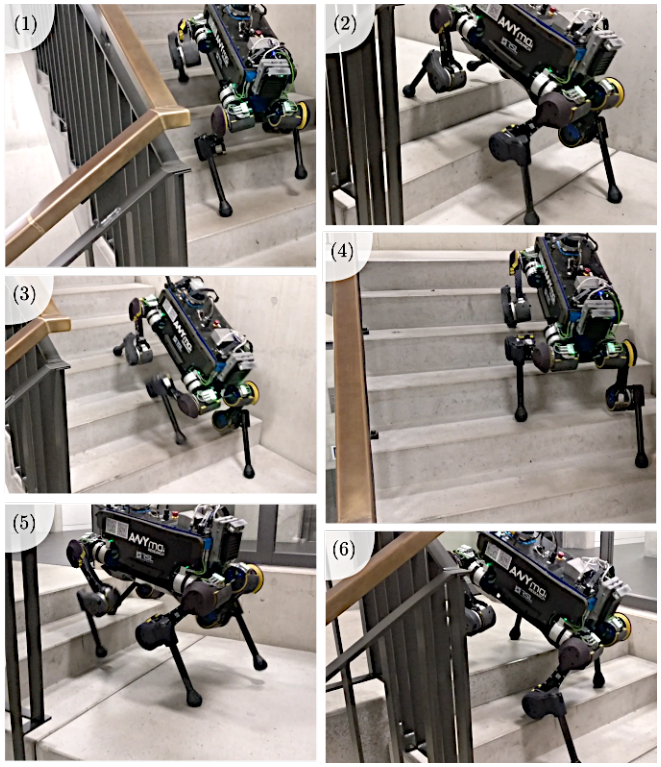

Fig. 8. ANYmal climbing up one floor on a stair case using perceptive trot. Each step is $0.17\,\mathrm{m}$ high and $0.29\,\mathrm{m}$ deep (36 degrees). The commanded heading velocity was $0.25\,\mathrm{m/s}$.

[4] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1096–1103.

[5] O. Magaña, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. Caldwell, and C. Semini, "Fast and continuous foothold adaptation for dynamic locomotion through cnns," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, April 2019.

[6] Y. Hong and B. Lee, "Real-time feasible footstep planning for bipedal robots in three-dimensional environments using particle swarm optimization," *IEEE/ASME Transactions on Mechatronics*, pp. 1–1, 2019.

[7] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 9–16.

[8] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3359–3365.

[9] C. Gehring, S. Coros, M. Hutler, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, and R. Siegwart, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics Automation Magazine*, vol. 23, no. 1, pp. 34–43, March 2016.

[10] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *International Conference on Climbing and Walking Robots*, 2014, pp. 433–440.

[11] 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 (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5. [Online]. Available: http://www.springer.com/de/book/9783319260525

[12] C. Gehring, C. D. Bellicoso, S. Coros, M. Bloesch, P. Fankhauser, M. Hutter, and R. Siegwart, "Dynamic trotting on slopes for quadrupedal robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 5129–5135.

[13] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, July 2018.

[14] R. Riener, M. Rabuffetti, and C. Frigo, "Stair ascent and descent at different inclinations," *Gait and posture*, vol. 15, pp. 32–44, 03 2002.

[15] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4170–4176, Oct 2019.

[16] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 38–44.

[17] R. Nishino and S. H. C. Loomis, "Cupy: A numpy-compatible library for nvidia gpu calculations," *31st confernce on neural information processing systems*, p. 151, 2017.