Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Driving in Challenging Terrain

Vivian S. Medeiros, Edo Jelavic, Marko Bjelonic, Roland Siegwart, Marco A. Meggiolaro, and Marco Hutter

***Abstract*—Wheeled-legged robots are an attractive solution for versatile locomotion in challenging terrain. They combine the speed and efficiency of wheels with the ability of legs to traverse challenging terrain. In this letter, we present a trajectory optimization formulation for wheeled-legged robots that optimizes over the base and wheels' positions and forces and takes into account the terrain information while computing the plans. This enables us to find optimal driving motions over challenging terrain. The robot is modeled as a single rigid-body, which allows us to plan complex motions and still keep a low computational complexity to solve the optimization quickly. The terrain map, together with the use of a stability constraint, allows the optimizer to generate feasible motions that cannot be discovered without taking the terrain information into account. The optimization is formulated as a Nonlinear Programming (NLP) problem and the reference motions are tracked by a hierarchical whole-body controller that computes the torque actuation commands for the robot. The trajectories have been experimentally verified on quadrupedal robot ANYmal equipped with non-steerable torque-controlled wheels. Our trajectory optimization framework enables wheeled quadrupedal robots to drive over challenging terrain, e.g., steps, slopes, stairs, while negotiating these obstacles with dynamic motions.**

***Index Terms*—Legged robots, wheeled robots, motion planning, optimization and optimal control.**

I. Introduction

LEGGED robots, such as ANYmal [1], have excellent mobility to cope with challenging terrain and are able to overcome large obstacles. Wheeled robots, on the other hand, exhibit high maneuverability on flat terrain, moving faster and more efficiently than legged systems. One attractive solution is to combine the advantages of both locomotion systems into a wheeled-legged system, that can cope with challenging environments at higher speeds [2]. Tasks where the execution time is important would greatly benefit from driving motions over challenging terrains, such as payload delivery or search and rescue. To this end, this work presents a Trajectory Optimization (TO) framework that generates dynamic driving motions for wheeled-legged quadrupedal robots with actuated wheels that allows the robot to overcome challenging obstacles by using the terrain map, as shown in Fig. 1.

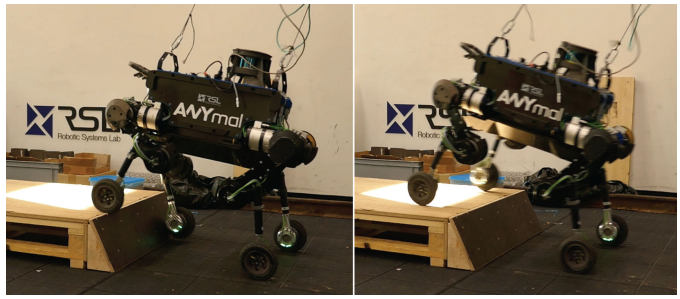

Fig. 1. The quadrupedal robot ANYmal equipped with four non-steerable, torque-controlled wheels, driving up a step 0.2 m high (over 40% of the legs length) and a 65° slope using our TO framework.

A. Related Work

In the field of wheeled-legged locomotion, most of the previous work focuses on robots performing motions in a purely reactive fashion. A number of authors have proposed reactive controller frameworks for wheeled-legged locomotion over uneven terrain. One example is extra-planetary rovers [3]–[7], that employ a purely reactive controller that can adapt to terrain variations by maintaining the desired base pose. These controllers are typically able to execute statically-stable[1] driving motions at low speeds, where the legs act as a sophisticated active suspension system. In [8] and [9], step-climbing strategies are presented

Manuscript received December 13, 2019; accepted April 11, 2020. Date of publication April 27, 2020; date of current version May 15, 2020. This letter was recommended for publication by Associate Editor K. Hashimoto and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by the Swiss National Science Foundation through the National Centres of Competence in Research Robotics and Digital Fabrication. This work has been conducted as part of ANYmal Research, a community to advance legged robotics. *(Corresponding author: Vivian Suzano Medeiros.)*

Vivian S. Medeiros and Marco A. Meggiolaro are with the Mechanical Engineering Department, PUC-Rio, Rio de Janeiro 22451-900, Brazil. Vivian was a Visiting Ph.D. Student at the Autonomous Systems Lab, ETH Zürich, at the time of this study. Her visit was funded by the Coordinating Agency for Advanced Training of Graduate Personnel (CAPES) – Brazil. (e-mail: viviansuzano@puc-rio.br; meggi@puc-rio.br).

Edo Jelavic, Marko Bjelonic, and Marco Hutter are with the Robotic Systems Lab, ETH Zürich 8092, Zürich, Switzerland (e-mail: jelavice@ethz.ch; marko-bjelonic@t-online.de; mahutter@ethz.ch).

Roland Siegwart is with the Autonomous Systems Lab, ETH Zürich 8092, Zürich, Switzerland (e-mail: rsiegwart@ethz.ch).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.2990720

[1]Statically stable locomotion requires the moving body to be stable at all times, which means that motion is not needed for maintaining balance. More specifically, the vertical projection of the center of gravity of the moving body will be contained within the convex hull of the body's points of contact with the ground at all times.

for wheeled-legged robots using wheel traction optimization and posture reconfiguration, but both limited to a quasi-static condition. In [9], only simulation results are presented.

The motion planning framework presented in [10]–[11] for the *CENTAURO* robot switches between walking and driving employing heuristics based on the terrain complexity. Experimental results show the robot overcoming obstacles like stones and steps with slow static maneuvers. Moreover, their approach does not consider solutions where the robot uses its wheels and legs simultaneously, which limits its ability to overcome obstacles compared to considering the whole-body in a single planning problem. In contrast, the motion planner presented by [12] solves the whole-body planning problem combining driving and stepping motions. The framework, however, focuses on generating kinematically feasible motions for heavy wheeled-legged vehicles performing slow maneuvers.

Recently, several approaches have been presented for dynamic motion generation with wheeled-legged systems. The wheeled bipedal robot *Handle* from Boston Dynamics [13] uses the wheels for driving over flat terrain and the legs for jumping over obstacles. The robot *Ascento* [14] presents a similar behavior, but a pre-defined jump motion is triggered by the user and no trajectory optimization is employed. A motion control and planning framework for ANYmal equipped with actuated wheels is presented in [15]–[16], in which the reference trajectories for the robot's center of mass (CoM) are continuously computed by a Zero-Moment Point (ZMP) optimization and are tracked by a hierarchical whole-body controller (WBC). Although this approach presented good experimental results, the planner uses a flat terrain assumption, which violates the validity of the ZMP model when moving over non-flat terrain and renders the approach not amenable for overcoming more challenging terrains, such as steep steps. Indeed, we tried to drive up steep slopes using the approach above with no success.

Hybrid driving-walking motions are shown for the ANYmal robot equipped with actuated wheels in [17], and for the robot *Robosimian* equipped with passive wheels in [18]. Both approaches have shown results only in simulations and over flat terrain. *Skaterbots* [19] uses a general TO framework for wheeled-legged robots that optimizes over several types of motions by solving a NLP problem, but the motions are performed only in flat terrain. All of the above approaches do not take into account terrain information and the flat terrain assumption makes them not very well suited for cases where the terrain contains steps or steep slopes.

B. Contribution

The main contribution with respect to the previous work is introducing a planning and control pipeline capable of negotiating rough terrain such as steep slopes with dynamic motions. On one hand, earlier work has presented navigation in challenging terrain, but only at low speeds (quasi-static conditions), less than $0.15\ \mathrm{ms}^{-1}$. On the other hand, other planning and control algorithms have produced dynamic motions, but employing a flat terrain assumption. In this work, we aim to bridge the gap between fast motions and motions in rough terrain. Our planning and control setup breaks the navigation in rough terrain problem into two subproblems: terrain aware motion planning and perceptive whole-body control. By adding the offline planning component we are able to execute faster motions over steep steps compared to previous approaches. Our algorithm is a TO framework for wheeled-legged robots that optimizes over the 6D base motion (position and orientation) as well as the wheels' positions and contact forces in a single planning problem, accounting for the terrain map and the robot's dynamics. This allows the robot to traverse a variety of challenging terrain, including large steps and drops, with dynamic driving motions that could not be generated without taking into account the terrain information. Moreover, our approach is general for all terrain types and the robot's base is not restricted to a desired height or orientation, which expands the range of achievable motions, especially for more complex terrains. Furthermore, we evaluated the proposed approach on ANYmal equipped with actuated non-steerable wheels in both simulations and real-world experiments. We show that ANYmal is able to traverse a variety of terrains, including steep inclinations 0.2 m high with 45° and 65° slopes at an average speed of $0.5\ \mathrm{ms}^{-1}$ and a maximum speed of $0.9\ \mathrm{ms}^{-1}$, which is over three times faster than previous approaches. To the best of our knowledge, such a fast negotiation of challenging obstacles using purely driving motions has not been shown before.

II. Trajectory Optimization

This section formalizes the TO problem for wheeled-legged robots and discusses its formulation as a NLP problem, as well as details of its collocation method. The goal of our motion planner is to solve an Optimal Control Problem (OCP) described as

$$\begin{aligned} \text{find} \quad & \boldsymbol{x}(t),\ \dot{\boldsymbol{x}}(t) \\ \text{subject to} \quad & \boldsymbol{x}(0) = \boldsymbol{x}_0, \quad \boldsymbol{x}(T) = \boldsymbol{x}_f, \\ & \boldsymbol{h}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \ddot{\boldsymbol{x}}(t)) \geq \mathbf{0}, \\ & \boldsymbol{g}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \ddot{\boldsymbol{x}}(t)) = \mathbf{0} \end{aligned}$$

where $\boldsymbol{x}(t)$ is the set of decision variables, given by the robot's CoM linear position and orientation (Euler angles), the wheels' contact positions and contact forces.

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{r}(t) & \boldsymbol{\theta}(t) & \boldsymbol{p}_i(t) & \boldsymbol{f}_i(t) \end{bmatrix}^T$$

The high-level user inputs are the initial and final state of the robot and the total time duration T of the trajectory. The duration T is defined based on the desired average speed for the robot's base.

In our approach, we employ a *Direct Collocation* method [20]-[21], where the OCP is transcribed into a NLP problem by optimizing over the decision variables in discrete times t_k sampled at a fixed interval ΔT along the trajectory, called *nodes*. All the optimization variables define each node at the time $t_k = k\Delta T$ of the trajectory, including the initial state, generating $n = \mathrm{floor}(T/\Delta T) + 1$ nodes. Each dimension of the variables is then represented in continuous time by connecting the nodes with third order polynomials, that can be fully defined by the value x and its derivative at the adjacent nodes and its time

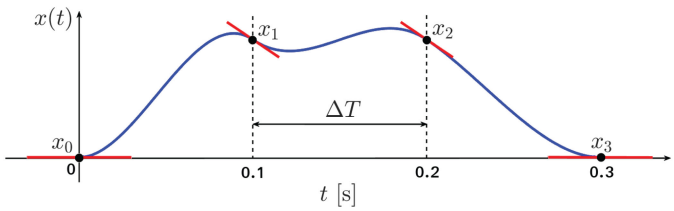

Fig. 2. Hermite spline parametrization. Decision variables inside the optimization are the black dots (nodes) and the red lines (state derivatives at each node). The blue curves are the polynomials defined by two consecutive nodes and their derivatives.

duration ΔT:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3,$$
$$a_i = f(x_k, \dot{x}_k, x_{k+1}, \dot{x}_{k+1}, \Delta T) \quad \forall\, k \in [0, n-1] \qquad (1)$$

where x_k is the state of the robot at the k^{th} node. This is called the Hermite parametrization, illustrated by Fig. 2, and allows to optimize over the state of the robot directly instead of the polynomials coefficients. This simplifies the formulation and the implementation of the NLP. The procedure to obtain the polynomial coefficients through the robot's state can be found in [22].

III. NLP Formulation

This section describes in detail the NLP problem solved to compute the motion plans, which formulation is summarized in Fig. 3. The right superscript denotes a component of the vector and the left superscript indicates the coordinate frame: I denotes the inertial frame, B denotes the base frame, W_i denotes the i^{th} wheel frame, located on the wheel's center and rotates with the wheel, and C_i denotes the i^{th} wheel contact frame, located on the wheels contact point on the ground. Fig. 4 shows the coordinate frames on the robot.

The formulation of the NLP was based on the one proposed in [22] for legged robots with point feet. In comparison with our formulation for wheeled robots performing driving motions, the main differences are that the velocity of the end-effector's contact point is no longer forced to zero and the rolling direction of the wheels is included to ensure consistency with wheeled locomotion. Additionally, since we focus on planning driving motions, the contact between the wheels and the ground is enforced during the entire trajectory.

A. Dynamic and Kinematic Constraints

The robot's dynamic model used for the TO is a Single Rigid Body model, in which the robot is approximated by a single rigid-body with mass and inertia located at the robots CoM. This model assumes that the mass of the legs is negligible compared to its base, which makes the dynamics of the robot independent of the joint configuration of the legs, keeping the formulation simpler and enabling faster convergence of the optimizer compared to the full-rigid body dynamics. This assumption is reasonable

find	${}^I\boldsymbol{r}(t) \in \mathbb{R}^3$	(CoM linear position)
	${}^I\boldsymbol{\theta}(t) \in \mathbb{R}^3$	(base Euler angles)
	for every wheel i :	
	${}^I\boldsymbol{p}_i(t) \in \mathbb{R}^3$	(wheels' motion)
	${}^I\boldsymbol{f}_i(t) \in \mathbb{R}^3$	(wheels' forces)
s.t.	$[{}^I\boldsymbol{r}, {}^I\boldsymbol{\theta}](0) = [{}^I\boldsymbol{r}_0, {}^I\boldsymbol{\theta}_0]$	(initial state)
	$[{}^I\boldsymbol{r}, {}^I\boldsymbol{\theta}](T) = [{}^I\boldsymbol{r}_g, {}^I\boldsymbol{\theta}_g]$	(goal state)
	$\boldsymbol{F}_d({}^I\boldsymbol{r}, {}^I\boldsymbol{\theta}, {}^I\boldsymbol{p}_i, {}^I\boldsymbol{f}_i) = \boldsymbol{0}$	(dynamic model)
	$s({}^I\boldsymbol{r}, {}^I\boldsymbol{\theta}, {}^I\boldsymbol{p}_i, {}^I\boldsymbol{f}_i) > \beta_{min}$	(stability measure)
	for every wheel i :	
	${}^I\boldsymbol{p}_i(t) \in \mathcal{R}_i({}^I\boldsymbol{r}(t), {}^I\boldsymbol{\theta}(t))$	(kinematic model)
	${}^I p_i^z(t) = h_{terrain}({}^I\boldsymbol{p}_i^{x,y}(t))$	(terrain height)
	${}^{C_i}\boldsymbol{f}_i^z(t) \geq 0$	(normal force)
	$\|\| {}^{C_i}\boldsymbol{f}_i^x(t) \|\| \leq f_{max}$	(maximum torque)
	${}^I\boldsymbol{f}_i(t) \in \mathcal{F}(\mu, \boldsymbol{n}, {}^I\boldsymbol{p}_i^{x,y}(t))$	(friction cone)
	${}^{C_i}\dot{p}_i^y(t) = 0$	(rolling constraint)
	for $k = 1..n$:	
	${}^I\ddot{\boldsymbol{p}}_i(t_k) - {}^I\ddot{\boldsymbol{p}}_i(t_{k-1}) = 0$	(acc continuity)

Fig. 3. Decision variables and constraints of our TO formulation. The use of a stability constraint and the constraints marked as blue are specific for optimizing driving motions over rough terrain.

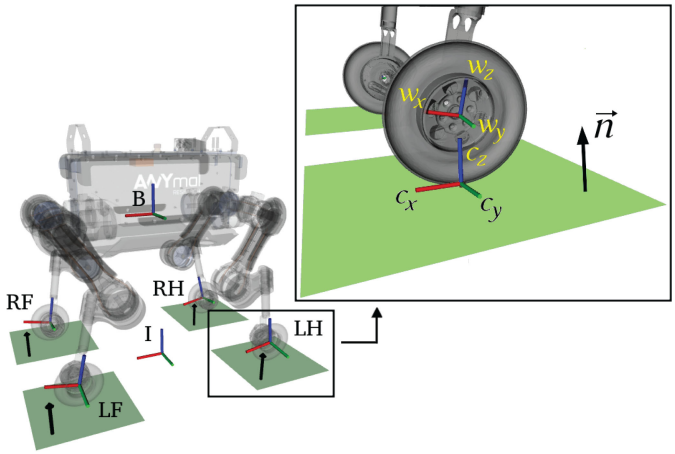

Fig. 4. Coordinate frames used for the motion planning. The frame B is attached to the CoM of the robot and each wheel has a frame attached to their center. On the right, there is a detailed view of the wheel's frame W_i and the contact frame C_i. The rotation axis of the wheels is the $\boldsymbol{w}_y$. The axis $\boldsymbol{c}_z$ on the contact frame is aligned with the terrain normal $\boldsymbol{n}$ and the $\boldsymbol{c}_x$ axis is aligned with the rolling direction of the wheel. In the wheels' frames, the first letter indicates the left (L) or right (R) and the second indicates front (F) or hind (H).

for most quadrupedal robots since a large portion of the mass is in the base of the robot. Each leg is up to an order of magnitude lighter than the base.

Using this simplified model, the robot's CoM linear acceleration $\ddot{\boldsymbol{r}}(t) \in \mathbb{R}^3$ is determined by the sum of all the contact forces on the wheels and the gravitational force; and the CoM

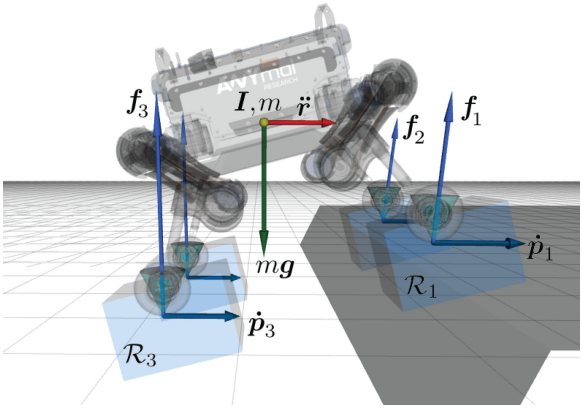

Fig. 5. The robot model for the optimization. The joint limits of the robot are assumed not violated if the wheel's contact point $\boldsymbol{p}_i$ is in inside the parallelepiped $\mathcal{R}_i$. The contact forces $\boldsymbol{f}_i$ on the wheels are constraint to remain inside the friction cone.

angular acceleration $\dot{\boldsymbol{\omega}}(t) \in \mathbb{R}^3$ is given by the Euler's rotation equation [23]. The dynamic equations are

$$m^I\ddot{\boldsymbol{r}}(t) = \sum_{i=1}^{4} {}^I\boldsymbol{f}_i(t) - m^I\boldsymbol{g}$$

$$\boldsymbol{I}^I\dot{\boldsymbol{\omega}}(t) + {}^I\boldsymbol{\omega}(t) \times \boldsymbol{I}^I\boldsymbol{\omega}(t) = \sum_{i=1}^{4} {}^I\boldsymbol{f}_i(t) \times ({}^I\boldsymbol{r}(t) - {}^I\boldsymbol{p}_i(t)) \tag{2}$$

where ${}^I\boldsymbol{\omega}(t) \in \mathbb{R}^3$ represents the angular velocity of the robot's base in the world frame, m denotes the robot's mass, ${}^I\boldsymbol{g} \in \mathbb{R}^3$ the gravity vector and $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ the inertia matrix of the robot, computed around the nominal stance position. The transformation from the Euler angles that define the orientation of the base (yaw, pitch, roll) to the angular velocities in world frame can be found in [23].

As for the kinematic constraints, we constrain the wheels positions to remain within a feasible workspace that moves together with the robot's base, approximated by a parallelepiped with fixed size located on the nominal position of the wheel relative to the robot's base, as depicted in Fig. 5. Such an approximation is common for robots with knee joints and fully articulated legs (at least 3 degrees-of-freedom per leg) [15], [22], [24]. The size of the parallelepiped must defined by the user by taking into account the position limits of the joints. The constraint is given by

$$-\boldsymbol{b} \;\leq\; \boldsymbol{R}_{BI}(\boldsymbol{\theta}(t))({}^I\boldsymbol{p}_i(t) - {}^I\boldsymbol{r}(t)) - {}^B\boldsymbol{p}_{in} \;\leq\; \boldsymbol{b}, \tag{3}$$

where $\boldsymbol{R}_{BI} \in \mathbb{R}^{3\times3}$ is the rotation matrix from the inertial frame to the base frame, ${}^B\boldsymbol{p}_{in} \in \mathbb{R}^3$ is the nominal position of the i^{th} wheel in the base frame and $\boldsymbol{b} = [b_x \quad b_y \quad b_z]^T$ is the vector of parallelepiped dimensions.

B. Wheels' Contact Constraints

For the entire trajectory, the robot is assumed to be in driving phase, i.e., the robot is not stepping, which imposes physical constraints on the wheels' motion and contact forces. Firstly, all the wheels must be in contact with the ground, which is enforced by

$${}^Ip_i^z(t) = h_{terrain}({}^I\boldsymbol{p}_i^{x,y}(t)) \tag{4}$$

where $h_{terrain}$ is the continuous 2.5D height map of the terrain [25].

Assuming constant contact also implies that the normal force on the wheels must always be positive. Since the contact forces are explicit decision variables, the forces can be directly constrained as

$${}^I\boldsymbol{f}_{n_i}(t) = {}^{C_i}\boldsymbol{f}_i^z(t) \geq 0, \tag{5}$$

where ${}^{C_i}\boldsymbol{f}_i^z(t)$ is the z component of the contact force on the i^{th} wheel expressed in the contact frame.

To ensure no slippage of the wheels' contact points, we constrain the tangential forces to remain inside the Coulomb friction cone defined by the terrain friction coefficient μ. In our implementation, the friction cone is approximated by a friction pyramid, which makes the constraint linear and thus, speeds up the computation. The constraint is given by

$$\begin{aligned} -\mu^I\boldsymbol{f}_{n_i}(t) &\leq {}^{C_i}\boldsymbol{f}_i^x(t) \leq \mu^I\boldsymbol{f}_{n_i}(t) \\ -\mu^I\boldsymbol{f}_{n_i}(t) &\leq {}^{C_i}\boldsymbol{f}_i^y(t) \leq \mu^I\boldsymbol{f}_{n_i}(t) \end{aligned} \tag{6}$$

Additionally, the traction forces are limited to a saturation value correspondent to the maximum torque of the wheel's motor, which is equivalent to limit the component of the contact force aligned with the rolling direction of the wheels:

$$-\tau_{\max}/w_r \leq {}^{C_i}\boldsymbol{f}_i^x(t) \leq \tau_{\max}/w_r, \tag{7}$$

Since we constrain the contact forces in a way that there is no slippage on the wheels, the maximum traction force is defined by the maximum torque on the wheel's motor $\tau_{\max} \in \mathbb{R}$ divided by the wheel's radius w_r.

Unlike for the robots with point feet, contact points for wheeled-legged robots can have a non-zero speed or acceleration. This introduces a rolling constraint, necessary to ensure the consistency of a driving motion. Motion of a wheel is consistent if the y-velocity of the wheels' contact point in the contact frame is zero:

$${}^{C_i}\dot{\boldsymbol{p}}_i^y(t) = 0 \tag{8}$$

The Hermite parametrization (Fig. 2) ensures continuously differentiable velocities and positions profiles for the base and the wheels. The accelerations, however, are not explicit decision variables and are computed from the derivative of the velocity polynomials, which allows for discontinuities in the acceleration profile. To avoid that, we constrain the wheel's accelerations to be equal at the polynomial junctions nodes, so there are no jumps in the wheels' acceleration that could cause jumps in the wheels' contact forces, which is not desired. The base's linear and angular accelerations are also constrained to be equal between the nodes, as they are in [22]. Finally, to prevent abrupt motions, the acceleration in the world frame on all wheels is limited to a maximum value.

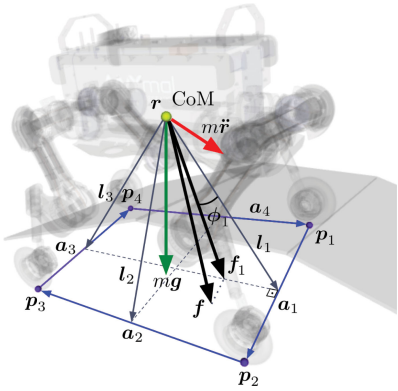

Fig. 6. Illustration of the Force-Angle Stability Measure for a quadrupedal robot, where $\boldsymbol{p}_i$ denotes the wheels' contact points; $\boldsymbol{a}_i$ denotes the tipover axes, defined as the vectors joining the contact points; $\boldsymbol{l}_i$ are the tipover axes normals, that intersect the tipover axis and the robot's CoM, which position is given by $\boldsymbol{r}$; $\boldsymbol{f}$ is the sum of all forces and angular loads acting on the CoM and $\boldsymbol{f}_1$ is the component of $\boldsymbol{f}$ that acts on the 1^{st} tipover axis; ϕ_1 is the stability angle w.r.t. to the first tipover axis. Same procedure is carried out to determine the correspondent $\boldsymbol{f}_i$ and ϕ_i for all tipover axes. All the vectors are represented in the inertial frame.

C. Stability Constraint

Since the contact forces are decision variables of our TO, it is possible to include a stability measure for the robot in the formulation. In this work, we define stability based on the measure proposed by [26], called the Force-Angle Stability Measure, illustrated for a quadrupedal robot in Fig. 6. The stability measure β is given by

$$\beta = \min(\phi_i), \quad i = 1, \ldots, 4, \tag{9}$$

Critical tipover stability occurs when β approaches zero, which happens when the force $\boldsymbol{f}_i$ is aligned with one of the tipover axes $\boldsymbol{l}_i$. If $\boldsymbol{f}$ lies outside the polygon defined by the contact points, β becomes negative and the robot starts to tip over. Hence, for the vehicle to remain in a stable condition, β must be positive. This measure is not limited to flat terrain and it can be applied to vehicles with any number of end-effectors, including manipulators. We incorporate the stability criterion into the TO as a constraint. The use of an objective function usually comes with higher computational effort and the amount of tuning parameters increases. Therefore, we limit ourselves to solving feasibility problems, similar to the approach presented in [22].

IV. Results

This section discusses the implementation and testing of several motions generated by our TO framework. We verify our planning and control pipeline in physical simulation on a variety of different terrains such as half-pipe terrain and stairs. We also perform experiments using ANYmal robot equipped with non-steerable torque-controlled wheels for steps with 45° and 65° slopes. All the motions are showed in details in the accompanying video, also available at https://youtu.be/DlJGFhGS3HM.

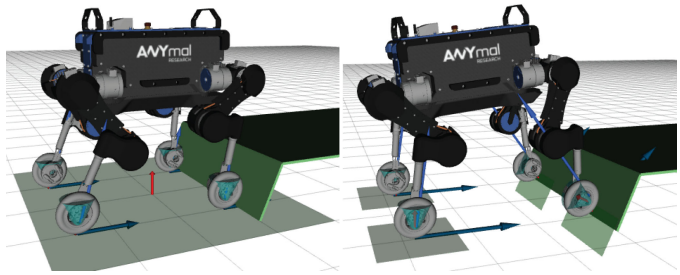

Fig. 7. Our extended WBC uses the terrain information to estimate the wheels contact points. (a) The approach described in [15] estimates the shape of the terrain by fitting a plane through the most recent wheels' positions. Note that the frictions cones are oriented with the normal plane, which is not accurate in this case. (b) Using the knowledge of the terrain normals, we are able to predict the contact points and the friction cones along the terrain map.

A. Implementation

The TO framework is implemented in C++ using the Ifopt [27] interface for the interior-point solver Ipopt [28]. The implementation is based on the software *TOWR* [22] for legged robots. The simulations are carried out in the robot simulation environment Gazebo with ODE [29] as the physics engine, using the full rigid body dynamics of the real ANYmal robot equipped with actuated wheels. Its legs feature three actuated joints arranged as a successive hip abduction/adduction, hip flexion/extension, and knee flexion/extension. The non-steerable torque controlled wheels are placed at the end of the legs.

The base and wheel motions are provided as input to the hierarchical WBC described in [15], that generates the actuation torques for the joints and the wheels while accounting for several constraints, such as actuator limits and friction cone constraints. We extend the existing controller (WBC) with the capability to leverage terrain normals. In [15] a unique plane is fitted through the most recent wheels' positions to estimate the terrain normal and the wheels' contact points. We extend that approach to use the knowledge of the terrain map to compute the terrain normal and contact point for each wheel separately. Since the friction cone axis is defined from the terrain normal, this mitigates incorrect estimation of the friction cones when traversing steep obstacles, as shown in Fig. 7. In this case, the lack of a terrain-aware motion or a contact force optimization prevents the robot from driving up the step, which is enabled by our framework.

The presented formulation requires a continuous 2.5D height map of the terrain. The height map can be either manually specified if the objects in the environment are known or generated from on-board sensors. Grid map representations such as *Octomap* [30] or a *Gridmap* [31] can be adapted to comply with the planner interface. Since our tests were carried out in known scenarios, the terrain map was analytically defined for each of the terrains.

For both simulations and experimental tests, the motion planner first reads the current state of the robot and then computes the trajectories for the desired time horizon. Fig. 8 gives an overview of the motion planning framework. The extended WBC tracks the reference trajectories, along with the robot state estimator, in

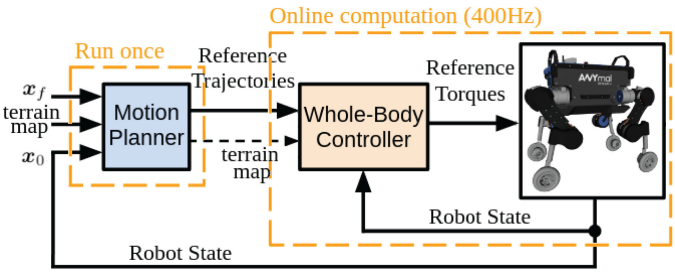

Fig. 8. Overview of the motion planning framework. The motion planner computes the reference trajectories for a specified time horizon that is given as input to the WBC, that computes the reference torques for the robot.

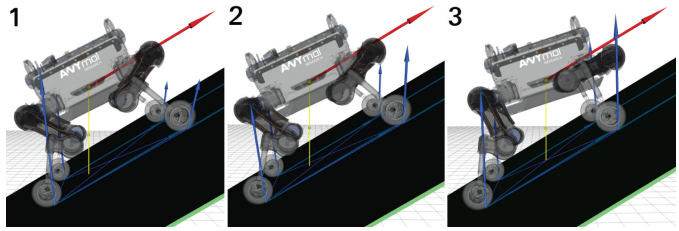

Fig. 9. ANYmal drives on a steep incline with a 30° inclination at average speed of 1.0 ms^{-1}. The light blue arrows on the wheels are the contact forces and the red arrow on the base is its linear velocity. Results for: (1) $\beta \geqslant 0$; (2) $\beta \geqslant 10°$; (3) $\beta \geqslant 20°$.

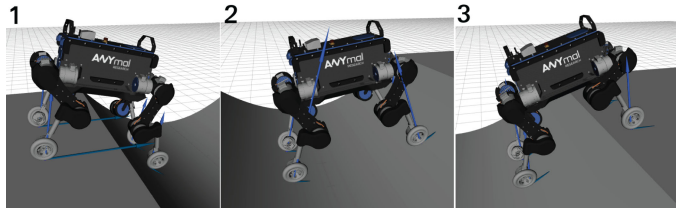

Fig. 10. The ANYmal robot driving over a 0.5 m half-pipe at an average speed of 1.2 ms^{-1}.

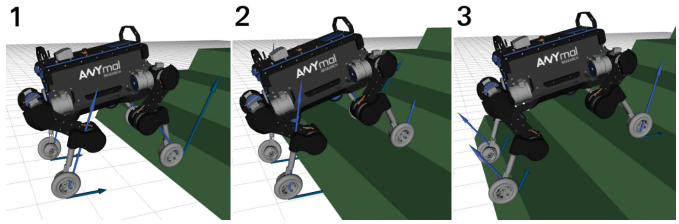

Fig. 11. ANYmal drives over stairs composed by steps with 0.2 m in height and a 0.4 m distance between them.

a 400 Hz loop. The state estimator module [32] fuses kinematic measurements from each actuator with the data from an inertial measurement unit (IMU).

For the motion planning, all the constraints are enforced in a time interval of 0.1 s, which is short enough to ensure physically feasible and dynamically consistent motions. Consequently, a trajectory with a 4.0 s time horizon has 2269 optimization variables, 1290 equality constraints and 2262 inequality constraints. The solver computation time depends on the complexity of the terrain and the optimization parameters, but remained in average 2.1 times[2] shorter than the planning horizon. All the derivatives of the constraints are provided analytically to the solver, which improves the solver's performance.

The position of the base is initialized by a linear interpolation between the initial and desired final position and the positions of the wheels are initialized assuming the default stance position of the robot's legs with respect to the base along the entire trajectory. The velocities on each node are initialized with the average speed of the robot, given by the total displacement of the trajectory divided by the total time duration.

B. Simulations

The simulations were carried out in the robot simulation environment Gazebo with different terrain types. We verify the effect of the tipover stability criterion introduced in Section III-C as well as the ability to traverse various challenging terrain. The effect of the stability constraint on the robot's motion is shown in Fig. 9, where snapshots of the robot driving on a slope with 30° inclination at an average speed of 1.0 ms^{-1} with different stability thresholds are shown. As expected, as the stability angle threshold increases, the trajectory presents a higher pitch angle for the base and the wheels position are adjusted to minimize the difference between the normal forces on the front and hind wheels. As the threshold increases, the size of the feasible solution set is reduced, the joint positions are closer to its limits and the computation cost for the optimization increases proportionally. As a compromise, the minimum stability angle was constrained to 10° for all trajectories.

Fig. 10 shows snapshots for the robot crossing a 0.5 m deep half-pipe at an average speed of 1.2 ms^{-1}. For this terrain, our TO takes 1.32 s to optimize the motion for a 3.0 s time horizon. Note that the robot maintains a kinematic feasible leg configuration by pitching the torso, which is only possible because we consider the whole body planning problem in our approach.

Our approach can handle multiple obstacles in succession such as stairs shown in Fig. 11. The stairs are composed by steps with 0.2 m in height, 0.4 m distance between them and transition with a 65° slope. A set of five steps is completed with an average speed of 0.5 ms^{-1}, which is faster than a legged robot with point feet could do. We achieve such a speed only because we compute plans for the robot before starting the maneuver which gives the controller more information for tracking compared to just reactive locomotion with flat terrain assumption. Trying to go up the steps with a reactive controller [15] has shown no success, even in simulation, as it can be seen in the accompanying video.

C. Experimental Results

We conducted experimental tests in four different configurations: two steps 0.2 m high (40% of the legs length) with a 45° and a 65° incline either in the center of the path or only on the right side of the robot. The optimization parameters were the

[2]The times stated in this work for the TO computation times were obtained on a 2.7 GHz dual-core Intel Core i7 laptop.

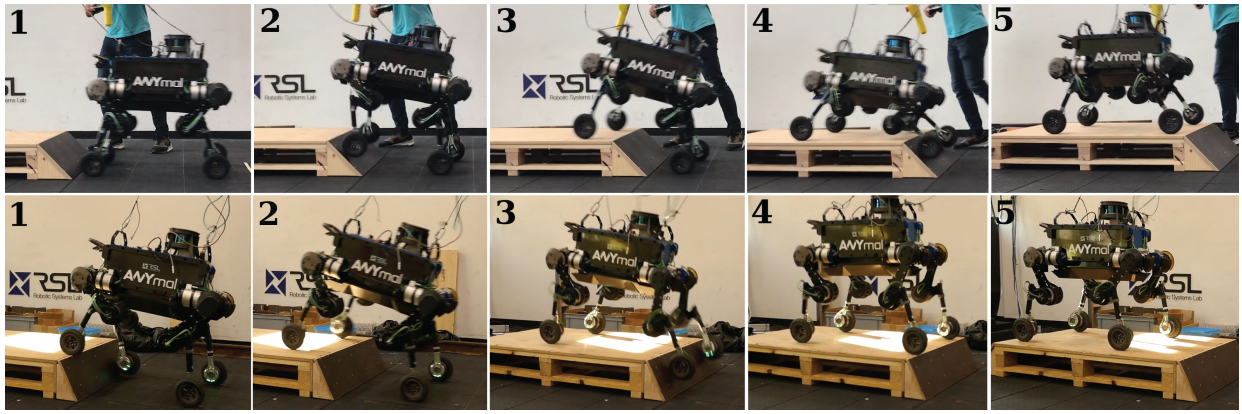

Fig. 12. Snapshots of the robot driving up a step 0.2 m high (40% of the legs length) using our TO framework. On the top row, ANYmal drives over the step with a 45° slope by driving both wheels up at the same time. On the bottom, the robot drives up a 65° slope by moving one front wheel at a time over the platform.

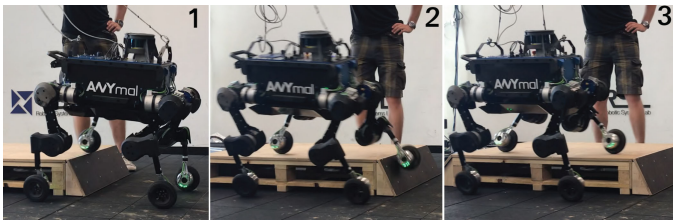

Fig. 13. The ANYmal robot drives up a platform with a 65° slope positioned in the right side of it's path.

same in all tests, with the exception of the robot's goal pose. All the obstacles are overcome with an average speed of 0.5 ms^{-1}.

Fig. 12 top row shows the robot driving over the step with a 45° slope moving both front wheels up at the same time. In this case, the base of the robot is kept in a lower position and with a higher distance between the front and the hind wheels to improve stability. On the bottom row, the robot successfully drives up a step with a 65° slope. For this motion, the robot drives up the left wheel first and, once both front wheels are on the platform, the hind wheels are driven up together to complete the motion. In this case, the transition over the step is more stable and the robot's CoM is maintained in a higher position. It is relevant to point out that the two different motions can only be obtained by providing an initial solution with a small shift between the left and right wheels' trajectories. This is due to the absence of a cost function in our formulation and the fact that the initial solution provided to the solver is a linear interpolation between the initial and final position of the wheels. In this case, the initial solution had the left wheels' positions shifted forwards 0.1 m in relation to the right wheels.

In Fig. 13, the platform with a 65° slope is now positioned on the right side of the robot, that drives up the obstacle in less than 4.0 s, showing that terrains with different heights in the $y-$direction are also successfully negotiated with our TO. To go up, the robot shifts and rolls its base to respect the kinematic limits. Fig. 14 depicts the desired motions compared with the measured positions obtained from the experiments for ANYmal driving up the 65° ramp in the center of its path. Our extended version of WBC is able to track the desired motions for the

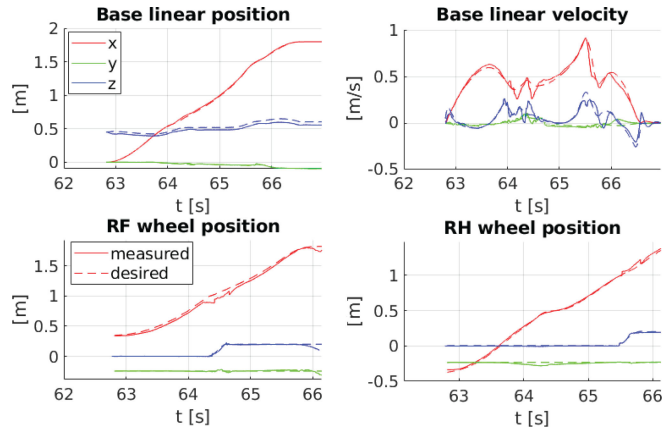

Fig. 14. The desired motions of the robot's base and wheels provided as input to the WBC (dashed line) and the measured positions (full line) obtained during the experiments for the 65° step.

base and the wheels with an average Root-Mean-Square-Error (RMSE) of 4 mm for the base and 15 mm for the wheels. The errors are computed using the robot's state computed by the state estimator module, which can present some drift if compared to ground truth measurements. Note that the base moves slower while the front wheels go up to maintain enough traction on the hind wheels; and moves faster when the hind wheels are going up, achieving a maximum speed of 0.88 ms^{-1}.

Considering the successful experiments with the 65° slope crossing, it is expected that the robot be able to traverse the stairs presented in Fig. 11. However, since the planning horizon is larger for such task, the lack of online adaptation of the trajectories and accumulated errors in the tracking controller could prevent the robot of climbing all the steps. One way to mitigate these problems is an implementation of the planner in a receding horizon fashion.

Generating base and wheels motions in a single planning problem using terrain information and without restrictions in speed or pose offer the advantage of an increased range of achievable motions for more complex terrains as a trade-off for the increased size of the optimization and its computational cost.

Relaxation of some constraints and a decrease in the number of variables would reduce the cost for the optimization, but could decrease its applicability.

V. Conclusion

We present a TO based planning and control pipeline that generates driving motions for wheeled quadrupedal robots, optimizing over the base motion and the wheels' positions, velocities and contact forces. Our framework consists of a terrain-aware planner and a terrain-aware controller that is an extension of the WBC presented in earlier work. Computing plans with terrain information enables us to generate driving motions over steep obstacles in a non-static manner, achieving an average speed of 0.5 ms^{-1} or higher. The feasibility of the trajectories are demonstrated in experiments with the ANYmal robot equipped with wheels driving over different terrains. The results show that we are able to achieve perceptive motion planning over multiple seconds horizons in challenging terrain.

Our trajectories are planned for a long time horizon in complex terrains which can be sensitive to drift and tracking error. In the future, we plan to investigate the implementation of our motion planner in a receding horizon fashion by using on-board state estimation of the robot. A major concern for making this possible is reducing the cost of the NLP solving. This could be achieved by employing smarter initialization strategies, similar to [33], that showed a significantly lower computational cost for a similar NLP formulation. Finally, we plan to extend the formulation to allow the generation of hybrid walking-driving motions including gait optimization, similar to [22].

References

[1] M. Hutter *et al.*, "Anymal - a highly mobile and dynamic quadrupedal robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 38–44.

[2] L. Bruzzone and G. Quaglia, "Review article: Locomotion systems for ground mobile robots in unstructured environments," *Mech. Sci.*, vol. 3, pp. 49–62, Jul. 2012.

[3] F. Cordes, F. Kirchner, and A. Babu, "Design and field testing of a rover with an actively articulated suspension system in a mars analog terrain," *J. Field Robot.*, vol. 35, pp. 1149–1181, 2018.

[4] K. Iagnemma and S. Dubowsky, "Traction control of wheeled robotic vehicles in rough terrain with application to planetary rovers," *I. J. Robot. Res.*, vol. 23, pp. 1029–1040, 10 2004.

[5] M. Lauria, Y. Piguet, and R. Siegwart, "Octopus—an autonomous wheeled climbing robot," in *Proc. 5th Int. Conf. Climb. Walk. Robots (CLAWAR)*, 2002, pp. 315–322.

[6] C. Grand, F. Benamar, and F. Plumet, "Motion kinematics analysis of wheeled–legged rover over 3d surface with posture adaptation," *Mechanism Mach. Theory*, vol. 45, no. 3, pp. 477–495, 2010.

[7] W. Reid, F. J. Prez-Grau, A. H. Gktoan, and S. Sukkarieh, "Actively articulated suspension for a wheel-on-leg rover operating on a martian analog surface," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 5596–5602.

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

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

[10] T. Klamt and S. Behnke, "Anytime hybrid driving-stepping locomotion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 4444–4451.

[11] T. Klamt *et al.*, "Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 1–8.

[12] E. Jelavic and M. Hutter, "Whole-body motion planning for walking excavators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2019, pp. 2292–2299.

[13] BostonDynamics, "Handle, mobile box handling robots for logistics." 2019. [Online]. Available: https://www.bostondynamics.com/handle

[14] V. Klemm *et al.*, "Ascento: A two-wheeled jumping robot," in *Proc. Int. Conf. Robot. Autom.*, May 2019, pp. 7515–7521.

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

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

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

[18] G. Bellegarda and K. Byl, "Trajectory optimization for a wheel-legged system for dynamic maneuvers that allow for wheel slip," in *Proc. IEEE Conf. Decis. Control*, 2019, pp. 7776-7781.

[19] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," in *Proc. ACM SIGGRAPH*, A. T. on Graphics (TOG), Ed., vol. 37. ACM, 2018, pp. 1–12.

[20] C. Hargraves and S. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *J. Guid., Control, Dyn.*, vol. 10, no. 4, pp. 338–342, Jul. 1987.

[21] 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.

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

[23] C. Gehring, C. D. Bellicoso, M. Bloesch, H. Sommer, P. Fankhauser, M. Hutter, and R. Siegwart, "Kindr library kinematics and dynamics for robotics." 2016. [Online]. Available: https://docs.leggedrobotics.com/kindr/cheatsheet_latest.pdf

[24] 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.

[25] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3019–3026, Oct. 2018.

[26] E. Papadopoulos and D. Rey, "The force-angle measure of tipover stability margin for mobile manipulators," *Vehicle Syst. Dyn. - VEH SYST DYN*, vol. 33, pp. 29–48, 01 2000.

[27] A. W. Winkler, "Ifopt - A modern, light-weight, Eigen-based C++ interface to Nonlinear Programming solvers Ipopt and Snopt." 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1135046

[28] 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, pp. 25–57, Mar. 2006.

[29] R. Smith, "Open dynamics engine," 2008, http://www.ode.org/. [Online]. Available: http://www.ode.org/

[30] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. [Online]. Available: http://octomap.github.com.

[31] 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

[32] M. Bloesch *et al.*, "State estimation for legged robots - consistent fusion of leg kinematics and IMU," in *Proc. Robot.: Sci. Syst.*, Sydney, Australia, Jul. 2012, pp. 17–24.

[33] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations," in *Proc. IEEE Int. Conf. Robotics Automat. (ICRA)*, 2020.