


Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots

Journal Article**Author(s):**

Bellicoso, C. Dario; Jenelten, Fabian; Gehring, Christian; [Hutter, Marco](#) 

Publication date:

2018-07

Permanent link:

<https://doi.org/10.3929/ethz-b-000221541>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Originally published in:

IEEE Robotics and Automation Letters 3(3), <https://doi.org/10.1109/LRA.2018.2794620>

Dynamic Locomotion through Online Nonlinear Motion Optimization for Quadrupedal Robots

C. Dario Bellicoso, Fabian Jenelten, Christian Gehring, Marco Hutter

Abstract—This paper presents a realtime motion planning and control method which enables a quadrupedal robot to execute dynamic gaits including trot, pace and dynamic lateral walk, as well as gaits with full flight phases such as jumping, pronging and running trot. The proposed method also enables smooth transitions between these gaits. Our approach relies on an online ZMP based motion planner which continuously updates the reference motion trajectory as a function of the contact schedule and the state of the robot. The reference footholds for each leg are obtained by solving a separate optimization problem. The resulting optimized motion plans are tracked by a hierarchical whole-body controller. Our framework has been tested in simulation and on ANYmal, a fully torque-controllable quadrupedal robot, both in simulation and on the actual robot.

Index Terms—Legged Robots, Optimization and Optimal Control

I. INTRODUCTION

ROBOTIC locomotion is a complex problem that involves a multi-body system interacting with the environment through multiple contact points. It becomes even more demanding when locomotion is executed in an unknown environment and under the presence of external disturbances. While animals have shown to be able to skillfully cope with these limitations, their robotic counterparts are still struggling to robustly locomote out of lab environments. Our goal is to have robots which are capable of safely executing dynamic locomotion over unstructured terrain. This would turn out to be extremely helpful in search and rescue scenarios, where tasks are carried out in an unpredictable environment and fast and stable locomotion can be essential. The capability of executing different kinds of gaits makes legged robots even more adaptable to different scenarios. A static walk might be the only option where careful stepping is the most critical requirement; running might be necessary if speed is the decisive factor for a successful mission; dynamic lateral walk might be the most energy efficient option at relatively low speeds. From these considerations, it emerges that the desired skills of a walking robot include not only the ability to execute

Manuscript received: September, 10, 2017; Revised December, 3, 2017; Accepted December, 8, 2017.

This paper was recommended for publication by Editor Paolo Rocco upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported in part by the Swiss National Science Foundation (SNF) through the National Centre of Competence in Research Robotics.

This work has been conducted as part of ANYmal Research, a community to advance legged robotics.

All authors are with the Robotic Systems Lab, ETH Zurich, Switzerland, {bellicoso, marco.hutter}@mavt.ethz.ch, fabianje@student.ethz.ch, gehrinch@ethz.ch

Digital Object Identifier (DOI): see top of this page.

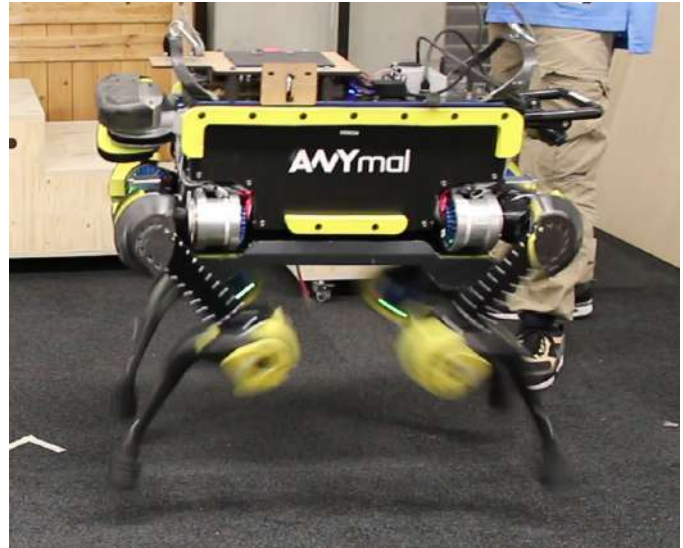


Fig. 1. ANYmal, a torque-controllable quadrupedal robot, executes a squat jump. The motion is computed by solving a nonlinear optimization problem which ensures dynamically stable plans.

dynamic gaits but also the ability to switch between the gaits. Although robots are not yet fully capable of autonomous locomotion in such environments, the past years have shown that both the academia and the industry are taking big steps in the development of tools and methods which would allow this.

The *Cheetah 2* [18], a quadrupedal robot developed at MIT, has demonstrated its agile skills by jumping over unexpected obstacles using a fast online *Model Predictive Controller* (MPC). The IIT *HyQ* [20] quadruped has shown walking using a creeping gait [16] over stepping stones [23]. In [22], the trajectory optimization framework is exploited to execute dynamic gaits such as trotting and pacing. Agile and dynamic skills have been demonstrated by the Boston Dynamics robots, including the quadrupedal robot *Spot* and the more recent *Spot Mini*, which additionally sports a robot arm for manipulation tasks. However, very little is known about the methods which enable such remarkable capabilities.

In [5], agile running motions have been demonstrated on the humanoid robot *Toro* [6]. Although the targeted robot platform is topologically different (i.e. humanoid versus quadruped), the problem formulation introduced there (e.g. parametrization of the motion with quintic splines; dynamic model used when in full flight phase) shares many similarities with ours. While impressive in the results, their approach is focused on the

planning and execution of running gaits on flat terrain or on stepping stones. We demonstrate, on the other hand, a single framework which deals with a larger variety of gaits, from a static walk up to a running trot and pace.

Over the past few years, *ANYmal* [12] (see Fig.1) has shown its ability to walk [1], trot [10] and climb over stairs [7]. In our previous work [2], we have advanced the capabilities of *ANYmal* by demonstrating dynamic lateral walking, pacing and gait transitions between these gaits. The method relied on a simplified and linearized *Zero-Moment Point* (ZMP, [21]) model to generate planar Center of Mass (CoM) motion plans, which were executed in an MPC-like fashion, relying on a hierarchical whole-body controller to track the reference motion. That approach, while novel, presented limitations both in the variety of gaits that could be produced and tracked, and in the simple approach which allowed us to switch between those gaits.

In this paper, we discuss our approach to overcome those limitations. First, we discuss the solution to an optimization problem which is solved online to generate motion plans for dynamic gaits such as trotting, walking, and pacing, as well as for more agile maneuvers such as jumping and running trot or pace; second, we introduce a separate online optimization problem which computes reference footholds; finally, we discuss a gait switching algorithm which allows us to switch between any of the gaits that are discussed in this paper. We show the results of our approach both in a simulated environment and on the quadrupedal robot *ANYmal*.

The frequency at which a motion plan is computed can vary depending on the type of gait and on the optimization parameters. As shown in our experiments, the optimization can be evaluated at more than 100Hz. The motion plans are tracked by a whole-body controller which runs at an update frequency of 400Hz.

II. MODEL FORMULATION

The model of a walking robot can be described as a free-floating base B to which limbs are attached. The motion of the entire system can be described w.r.t. a fixed inertial frame I . The position of the Base w.r.t. the inertial frame is written as ${}^I\mathbf{r}_{IB} \in \mathbb{R}^3$. The orientation $\mathbf{q}_{IB} \in SO(3)$ of the Base w.r.t. the inertial frame is parametrized using Hamiltonian unit quaternions. The limb joint angles are stacked in the vector $\mathbf{q}_j \in \mathbb{R}^{n_j}$. We write the generalized coordinates vector \mathbf{q} and the generalized velocities vector \mathbf{u} as

$$\mathbf{q} = \begin{bmatrix} {}^I\mathbf{r}_{IB} \\ \mathbf{q}_{IB} \\ \mathbf{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \mathbf{u} = \begin{bmatrix} {}^I\mathbf{v}_B \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\mathbf{q}}_j \end{bmatrix} \in \mathbb{R}^{n_u}, \quad (1)$$

where $n_u = 6 + n_j$, ${}^I\mathbf{v}_B$ and ${}_B\boldsymbol{\omega}_{IB}$ are the linear and angular velocity of the Base w.r.t. the inertial frame expressed respectively in the I and B frame. The equations of motion of mechanical systems which are in contact with the environment can be written as $\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{h}(\mathbf{q}, \mathbf{u}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_s^T\boldsymbol{\lambda}$, where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_u \times n_u}$ is the mass matrix and $\mathbf{h}(\mathbf{q}, \mathbf{u}) \in \mathbb{R}^{n_u}$ is the vector of Coriolis, centrifugal and gravity terms. The selection matrix $\mathbf{S} = \begin{bmatrix} \mathbf{0}_{n_\tau \times (n_u - n_\tau)} & \mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix}$ selects which Degrees of Freedom (DoFs) are actuated. If all limb joints are

actuated, then $n_\tau = n_j$. The vector of constraint forces $\boldsymbol{\lambda}$ is mapped to the joint space torques through the support Jacobian $\mathbf{J}_s \in \mathbb{R}^{3n_c \times n_u}$, which is obtained by stacking the constraint Jacobians as $\mathbf{J}_s = [\mathbf{J}_{C_1}^T \cdots \mathbf{J}_{C_{n_c}}^T]^T$, with n_c the number of limbs in contact.

III. MOTION OPTIMIZATION

Our motion planning framework consists of two main modules, namely a CoM motion planner and a foothold generator, both of which are separately solving an optimization problem. The computation is carried out in parallel to the the main control loop, such that it does not interrupt the whole-body motion tracking controller. Motion plans for the swing feet are obtained by generating splines through the current position of the foot and its reference optimized foothold.

The motion plans are computed in the *Plan frame* P , which is located at the footprint center projected onto the local terrain along the terrain normal. The footprint is represented by the location of the feet of the robot. The orientation of the Plan frame is such that it is parallel to the local estimation of the terrain from the contact points[9], while its yaw angle is computed s.t. the x axis is aligned with the reference high-level velocity. The latter is generated from an external source, e.g. an operator device or a high-level navigation planner. The Plan frame allows the optimization parameters (e.g. weights relative to a specific coordinate to the CoM) to be independent of the inertial frame and to be properly defined w.r.t. the desired direction of motion.

To drive locomotion to the reference speed, footholds are generated (section III-C) for all legs such that the robot can achieve the desired high-level velocity on average. This information, together with the contact schedule (i.e. the predefined lift-off and touch-down timings for each leg), is used to generate a sequence of support polygons (section III-D) which are sent to the motion plan optimizer (section III-A). This in turn will produce position, velocity and acceleration profiles for the whole-body center of mass. Fig.2 depicts an overview of the entire motion planning and tracking framework.

We build on our previous work [2] which described how to produce a motion plan for the x and y coordinates of the Center of Mass (CoM) which can be tracked by a quadrupedal robot. In that case, the reference height of the CoM was set to a user-specified value and tracked by the controller. To optimize for gaits which include full flight phases, the optimizer must know how to produce motion plans which will be projected to contact forces which produce jumping motions. To do this, we include the z coordinate of the CoM in the optimization, such that we optimize for the position, velocity and acceleration of the CoM of the system. This has important implications on the problem formulation. First, the dimension of the optimization problem is increased. Then, by dropping the assumption that the z component of the center of mass acceleration $\ddot{\mathbf{p}}_{CoM}$ is zero, constraints on the ZMP model become nonlinear. Although the method described in this paper is more complex (thus, more computationally expensive) than the one that we build on, we are still capable of computing the plans online and executing them in a MPC-like fashion, i.e. we continuously

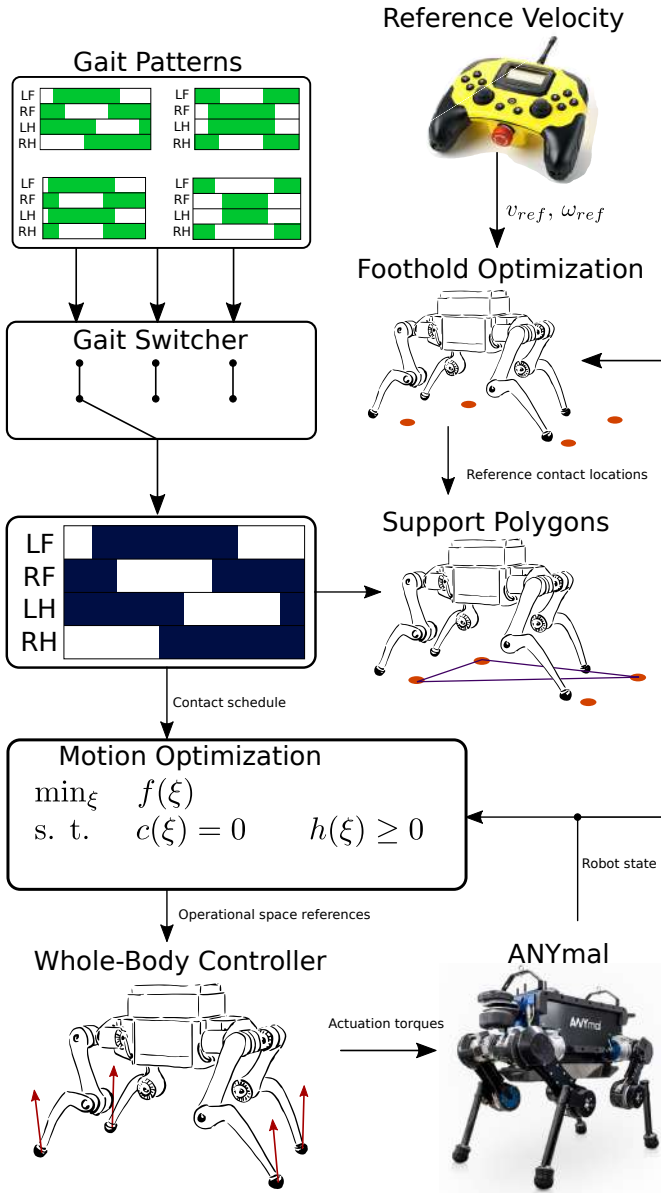


Fig. 2. An overview of the planning and control architecture described in this paper.

run the motion optimizer initialized with the most recent state of the robot to update the motion plan that is tracked by the whole-body motion controller.

A. Problem formulation

We describe each coordinate of the CoM motion plan as a sequence of quintic splines. As it will be explained in detail in the following sections, the motion plan that we are optimizing is described over a sequence of support polygons which depend on the type of gait that is being executed. In our implementation, for each component of the CoM there is at least one spline for each support polygon. Two splines are used for full flight phases. The motion of one of the coordinates of the CoM, for example the x component, can be described by

the i -th spline as

$$\begin{aligned} x(t) &= \alpha_{i5}^x t^5 + \alpha_{i4}^x t^4 + \alpha_{i3}^x t^3 + \alpha_{i2}^x t^2 + \alpha_{i1}^x t + \alpha_{i0}^x \\ &= [t^5 \ t^4 \ t^3 \ t^2 \ t \ 1] \\ &\quad \cdot [\alpha_{i5}^x \ \alpha_{i4}^x \ \alpha_{i3}^x \ \alpha_{i2}^x \ \alpha_{i1}^x \ \alpha_{i0}^x]^T \\ &= \boldsymbol{\eta}^T(t) \boldsymbol{\alpha}_i^x, \end{aligned} \quad (2)$$

with $t \in [\bar{t}, \bar{t} + \Delta t_i]$, where \bar{t} is the sum of time durations of all splines up to the $(i-1)$ -th one, and Δt_i is the time duration of the i -th spline.

Using the notation introduced, we can compactly write velocity and acceleration as

$$\dot{x}(t) = \dot{\boldsymbol{\eta}}^T(t) \boldsymbol{\alpha}_i^x \quad \ddot{x}(t) = \ddot{\boldsymbol{\eta}}^T(t) \boldsymbol{\alpha}_i^x, \quad (3)$$

where we have used the time vectors

$$\begin{aligned} \dot{\boldsymbol{\eta}}(t) &= [5t^4 \ 4t^3 \ 3t^2 \ 2t \ 1 \ 0]^T \\ \ddot{\boldsymbol{\eta}}(t) &= [20t^3 \ 12t^2 \ 6t \ 2 \ 0 \ 0]^T. \end{aligned} \quad (4)$$

Similar considerations can be carried out for the y and z components of the CoM. The vector of optimization parameters is then defined as $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_0^T \ \dots \ \boldsymbol{\alpha}_i^T \ \dots \ \boldsymbol{\alpha}_{n_s}^T]^T$, with $3n_s$ the total number of splines and $\boldsymbol{\alpha}_i = [\boldsymbol{\alpha}_i^x \ \boldsymbol{\alpha}_i^y \ \boldsymbol{\alpha}_i^z]^T$. By also defining

$$\mathbf{T}(t) = \begin{bmatrix} \boldsymbol{\eta}^T(t) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \dot{\boldsymbol{\eta}}^T(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddot{\boldsymbol{\eta}}^T(t) \end{bmatrix}, \quad (5)$$

we can easily write the CoM position $\mathbf{p}_{CoM}(t) = \mathbf{T}(t) \boldsymbol{\alpha}_i$, where $\mathbf{p}_{CoM} \in \mathbb{R}^3$. Velocity and acceleration vectors can be written as $\dot{\mathbf{p}}_{CoM}(t) = \dot{\mathbf{T}}(t) \boldsymbol{\alpha}_i$ and $\ddot{\mathbf{p}}_{CoM}(t) = \ddot{\mathbf{T}}(t) \boldsymbol{\alpha}_i$ respectively.

B. Center of mass optimization

Due to the nature of the constraints that will be described in the following sections, we formulate the motion planning algorithm as a nonlinear optimization problem which minimizes a generic cost function $f(\boldsymbol{\xi})$ subject to equality and inequality constraints $\mathbf{c}(\boldsymbol{\xi}) \leq \mathbf{0}$ and $\mathbf{h}(\boldsymbol{\xi}) = \mathbf{0}$. To numerically solve the optimization problem, we use the *Sequential Quadratic Programming* (SQP) algorithm [17], which requires the computation of Jacobians and Hessians of the constraints and the cost function. The optimization, continuously re-evaluated as soon as the previous one is completed, provides a motion plan over a time horizon of τ seconds, where τ is the periodicity of the locomotion gait (e.g. walk or trot) that is being optimized for.

In the following, we describe each single contribution to the cost function and the setup of all constraints used in our formulation.

1) *Cost function*: As done in [23] and [2], we minimize the acceleration of the entire motion plan. To do this, we set a quadratic cost computed for spline segment as

$$\mathbf{Q}_k^{acc} = \begin{bmatrix} (400/7)\Delta t_k^7 & 40\Delta t_k^6 & 24\Delta t_k^5 & 10\Delta t_k^4 & & \\ 40\Delta t_k^6 & 28.8\Delta t_k^5 & 18\Delta t_k^4 & 8\Delta t_k^3 & & \\ 24\Delta t_k^5 & 18\Delta t_k^4 & 12\Delta t_k^3 & 6\Delta t_k^2 & & \\ 10\Delta t_k^4 & 8\Delta t_k^3 & 6\Delta t_k^2 & 4\Delta t_k & & \\ & \mathbf{0}_{2 \times 4} & & & & \\ & & & & & \mathbf{0}_{2 \times 2} \end{bmatrix} \quad (6)$$

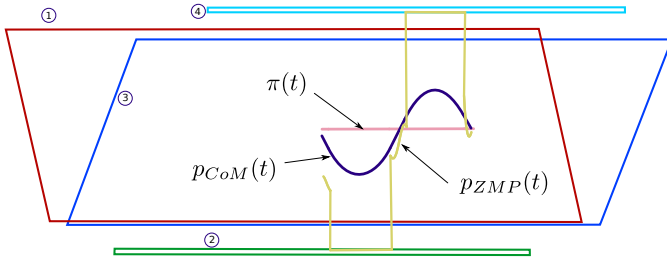


Fig. 3. The top view of a motion plan computed for a pace (both lateral legs simultaneously swinging). The blue curve is the desired CoM position, the yellow curve the resulting ZMP position, and the pink curve is the path regularizer. The support polygon sequence is numbered from 1 to 4.

and $\mathbf{c}_k^{acc} = \mathbf{0}$, with Δt_k the time duration of the k -th spline segment in seconds. As explained in [13], the Hessian matrix \mathbf{Q}_k^{acc} in eq.(6) is obtained by squaring and integrating the acceleration of the CoM over the time duration of the k -th spline. \mathbf{Q}_k^{acc} is added as a sub-matrix to the Hessian of the overall cost function.

Soft constraints are set to impose constraints on the initial and final state of the whole motion plan. The initial acceleration is read from that of the previous optimization. The final desired position is computed as a function of the reference high-level linear velocity \mathbf{v}_{ref} and the z -component of the angular velocity ω_{ref} and the optimization horizon τ . We assume that these are constant over the optimization horizon (which is a valid assumption if they change at a much slower rate than the optimization update frequency), hence we can compute the final position as

$$\mathbf{p}_{final} = \mathbf{p}_{init} + \mathbf{R}(\tau\hat{\omega}_z)(\tau\mathbf{v}_{ref}), \quad (7)$$

where the second term is the rotation of a position vector computed as a velocity dependent offset vector and $\hat{\omega}_z = [0 \ 0 \ \omega_{refz}]^T$.

External disturbances and continuous updates of the motion plan can cause a drift of the floating base of the robot w.r.t. the reference footholds over time. For this reason, we minimize the deviation between the motion plan w.r.t. to a *path regularizer* π (see Fig.3) which represents a high-level approximation of the motion plan. The initial position of π is computed as the footprint center averaged over the optimization horizon τ , where it is assumed that the feet are either in contact or move with a constant velocity. The final position is computed through the reference high-level velocity as in (7). We set the initial velocity equal to the reference velocity \mathbf{v}_{ref} , and the final one to be aligned with the predicted torso orientation at the end of the motion plan. The initial and final accelerations of π are set to zero. The initial and final height of π are set to a user specified reference value.

Typically, the motion plan shows large deviations w.r.t. the path regularizer. Overshoots in the vertical direction can cause violation of kinematic limits in the legs. To limit this issue, we bound overshoots along the z axis by adding a cost in the form $\lambda_{lin}\epsilon_z + \lambda_{quad}\epsilon_z^2$ and augmenting our formulation with

the constraints

$$\begin{aligned} \mathbf{p}_{CoM}^z(t) - \pi_z(t) &\leq \epsilon_z \\ \mathbf{p}_{CoM}^z(t) - \pi_z(t) &\geq -\epsilon_z \\ \epsilon_z &\geq 0, \end{aligned} \quad (8)$$

where ϵ is a slack variable which is added to the vector of optimization variables ξ . Due to the time dependency, the trajectory needs to be sampled where each sample point introduces two additional inequality constraints.

2) *Equality constraints*: To ensure that each pair of adjacent splines is connected, we set junction constraints. Using the notation introduced in section III-A, we write the junction constraints for the x coordinate between the k -th and $(k+1)$ -th spline as

$$\begin{bmatrix} \boldsymbol{\eta}(t_{fk})^T & -\boldsymbol{\eta}(0)^T \\ \dot{\boldsymbol{\eta}}(t_{fk})^T & -\dot{\boldsymbol{\eta}}(0)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_k^x \\ \boldsymbol{\alpha}_{k+1}^x \end{bmatrix} = \mathbf{0}, \quad (9)$$

with t_{fk} representing the duration in seconds of spline s_k . Similarly, (9) can be written for the y and z coordinates.

Junction conditions are formulated differently if two spline segments are connected through a *full flight phase*. In this case, the dynamics of the CoM are described by $\dot{\mathbf{p}}_{CoM} = \mathbf{g}$, where \mathbf{g} is the gravity acceleration vector. Integrating the dynamics yields

$$\begin{aligned} \dot{\mathbf{p}}(t) &= \mathbf{g}t + \dot{\mathbf{p}}_{CoM}(0) \\ \mathbf{p}(t) &= \frac{1}{2}\mathbf{g}t^2 + \dot{\mathbf{p}}_{CoM}(0)t + \mathbf{p}_{CoM}(0). \end{aligned} \quad (10)$$

The constraints at touch down can be written as a function of the spline coefficients as

$$\begin{aligned} \ddot{\mathbf{T}}(0)\boldsymbol{\alpha}_{i+1} &= \mathbf{g} \\ \dot{\mathbf{T}}(0)\boldsymbol{\alpha}_{i+1} &= \mathbf{g}t_f + \dot{\mathbf{T}}(t_i)\boldsymbol{\alpha}_i \\ \mathbf{T}(0)\boldsymbol{\alpha}_{i+1} &= \frac{1}{2}\mathbf{g}t_f^2 + \left[\dot{\mathbf{T}}(t_i)t_f + \mathbf{T}(t_i) \right] \boldsymbol{\alpha}_i \end{aligned} \quad (11)$$

with t_i indicating the duration of the i spline. Similar replacements can be found for initial and final conditions if the first or the last support polygon corresponds to a full flight phase.

3) *Inequality constraints*: As shown in [19], the ZMP position \mathbf{p}_{ZMP} w.r.t. the origin of the plan frame O is described by

$$\mathbf{p}_{ZMP} = \frac{\mathbf{n} \times \mathbf{M}_O^{gi}}{\mathbf{n}^T \mathbf{F}^{gi}}, \quad (12)$$

where \mathbf{M}_O^{gi} and \mathbf{F}^{gi} are the components of the so-called *gravito-inertial wrench*[4], which are computed as

$$\begin{aligned} \mathbf{M}_O^{gi} &= m \cdot \mathbf{p}_{CoM} \times (\mathbf{g} - \ddot{\mathbf{p}}_{CoM}) - \dot{\mathbf{L}} \\ \mathbf{F}^{gi} &= m \cdot \mathbf{g} - \dot{\mathbf{P}}, \end{aligned} \quad (13)$$

with m being the mass and \mathbf{P} and \mathbf{L} the linear and angular momentum at the CoM. In the following, we will approximate $\dot{\mathbf{L}} = \mathbf{0}$ as we do not optimize for rotations and their derivatives.

To ensure dynamic stability of the planned motions, we constrain the ZMP to always lie in the support polygon. To do this, we write constraints in the form

$$\mathbf{h}_{ZMP} = \mathbf{d}^T \mathbf{p}_{ZMP} + r \geq 0, \quad (14)$$

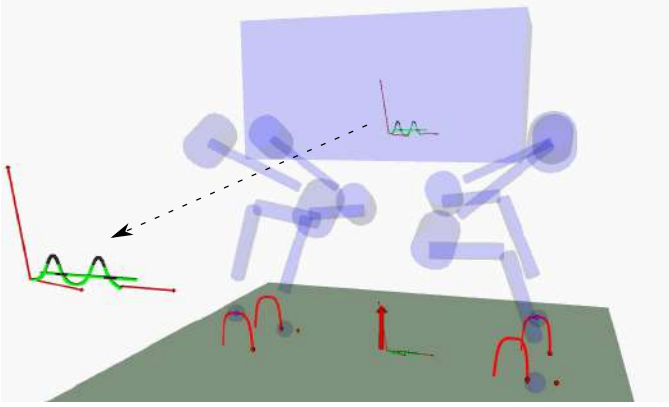


Fig. 4. A motion plan computed for a running trot, which includes a full flight phase. The computed motion plan (shown in close-up) sets vertical accelerations which cause the robot to jump.

where $\mathbf{d}^T = [p \ q \ 0]$ and r are the coefficients which describe an edge that belongs to the support polygon. Substituting (12) into (14) we get

$$\begin{aligned} & \mathbf{d}^T \mathbf{S}(\mathbf{n}) \mathbf{M}_O^{g_i} + r \mathbf{n}^T \mathbf{F}^{g_i} \\ & = \mathbf{d}^T \mathbf{S}(\mathbf{n}) \mathbf{S}(\mathbf{p}_{CoM}) (\mathbf{g} - \ddot{\mathbf{p}}_{CoM}) + r \mathbf{n}^T (\mathbf{g} - \ddot{\mathbf{p}}_{CoM}) \geq \mathbf{0} \end{aligned} \quad (15)$$

where $\mathbf{S}(\mathbf{a})$ is a skew symmetric matrix computed such that $\mathbf{S}(\mathbf{a})\mathbf{b} = \mathbf{a} \times \mathbf{b}$.

Using (15), we compute the gradient $\nabla \mathbf{h}_{ZMP}$ of (14) w.r.t. the position and acceleration of the com $\mathbf{p}_{CoM}, \ddot{\mathbf{p}}_{CoM}$ by writing

$$\nabla \mathbf{h}_{ZMP} = \begin{bmatrix} \mathbf{\Gamma} \cdot (\ddot{\mathbf{p}}_{CoM} - \mathbf{g}) \\ -\mathbf{\Gamma} \cdot \mathbf{p}_{CoM} - r \mathbf{n} \end{bmatrix} \quad (16)$$

with $\mathbf{\Gamma} = \mathbf{S}(\mathbf{S}^T(\mathbf{n})\mathbf{d})$, and the Hessian $\nabla^2 \mathbf{h}_{ZMP}$ as

$$\nabla^2 \mathbf{h}_{ZMP} = \begin{bmatrix} \mathbf{0} & \mathbf{\Gamma}^T \\ -\mathbf{\Gamma}^T & \mathbf{0} \end{bmatrix}, \quad (17)$$

The Hessian in (17) is anti-symmetric, hence it drops out from the optimization.

We soften the inequality constraints of the initial n_{ineq} samples, with n_{ineq} a tuning parameter set by the user. To achieve this, we add slack variables ϵ_{ineq} to the optimization parameters ξ . The formulation is modified by adding the cost $\lambda_{lin} \epsilon_{ineq} + \lambda_{quad} \epsilon_{ineq}^2$ and appending the constraints $\mathbf{c}_{ineq} \geq -\epsilon_{ineq}, \quad \epsilon_{ineq} \geq 0$, where \mathbf{c}_{ineq} are the first n_{ineq} constraints described in (15). From a physical point of view the relaxation amounts to a variable sized support polygon that cannot be smaller than the nominal one.

4) *Assigning a new plan*: When a new motion plan is obtained and sent to the controller, it is important to properly append it to the previously executed plan. To do this, we first store the computation time t_c that was necessary for the solver to optimize. We use this as an initial guess to search for the position in the motion plan that is closest to the current measured one, such that the transition to the new plan is a smooth one. To do this, we solve a line search by writing

$$t = \arg \min_{\mathbf{w}} \|\mathbf{p}(t) - \mathbf{p}_{meas}\|_2^2 \quad (18)$$

where \mathbf{W} is a diagonal positive definite weighting matrix and \mathbf{p}_{meas} is the measured CoM position after completing the optimization. We use a Newton method with a back-tracing line search algorithm that usually converges within 3 iterations and leads to an average correction of 20% of the initial spline time guess. During experiments with ANYmal, it has been found that a line search significantly increases the control performance.

C. Foothold optimization

In our previous works ([10], [2]) we have computed reference footholds using a linear inverted pendulum model [10] which predicts the next foothold as a function of the measured and desired velocity of the torso. This method has shown to add robustness to the real system in case of external disturbances or deviation from the reference motion. However, we wish to embed it into a framework which allows us to take into account different contributions to the computation of the footholds, as well as to set constraints on the final result. For this reason, we set up a Quadratic Programming (QP) problem as

$$\min_{\xi} \frac{1}{2} \xi^T \mathbf{Q} \xi + \mathbf{c}^T \xi \quad \text{s. t.} \quad \mathbf{D} \xi \leq \mathbf{f}. \quad (19)$$

where $\xi \in \mathbb{R}^{2n_{feet}}$ is a vector of x and y components of footholds \mathbf{p}_{f_i} , with $i = 1, \dots, n_{feet}$, and $n_{feet} = 4$ the total number of feet. Similarly to what is done for the CoM motion planner, we optimize in parallel to the main control loop. Thus we update the *foothold plan* whenever a new optimization is ready. In the following we describe each contribution to the cost function in (19) as well as the constraints which appear in it.

1) *Cost function*: A user-defined default foot position can be assigned which is relative to a default standing configuration. This can be interpreted as a regularization term in the foothold optimization. To track these default foothold locations \mathbf{p}_{ref_i} , we add

$$\mathbf{Q}_{def_i} = \mathbf{W}_{def}, \quad \mathbf{c}_{def_i} = -\mathbf{W}_{def}^T \mathbf{p}_{ref_i} \quad (20)$$

to the cost function described in (19). The choice of the default footholds will influence how wide the footprint will be when all legs are in contact with the environment. While this can make a trotting gait more robust to disturbances, it will produce wider lateral motions in slower walking gaits. In practice, the default foothold location that worked reliably during our experiments on ANYmal was computed as the vertical projection of the hips to the terrain.

To track on average high-level velocities which drive the whole locomotion framework, we penalize deviations from foothold locations. These are computed assuming that a constant velocity is achieved over the duration of the optimization horizon. To avoid jumps in the reference footholds, we additionally set a cost to the distance between the current solution and the previously computed one.

Finally, we add to the cost a stabilizing term which is a function of the inverted pendulum model as described in [10], which computes the k -th foothold as $w(\mathbf{v}_{ref} - \mathbf{v}_{hip_k}) \sqrt{h/g}$, with w a positive scalar weight, \mathbf{v}_{ref} the high level reference

velocity, \mathbf{v}_{hip_k} the linear velocity of the k -th hip, h the height of the k -th hip from the ground and g the gravity acceleration constant.

2) *Inequality constraints*: To avoid computing footholds which would violate the kinematic extension of the legs, we exploit the QP setup by adding inequality constraints on the feasible location of each foothold. We do this by considering the maximum leg extension l_{max} and the measured height of each hip h_i . By projecting the hip location on the terrain \mathbf{h}_0 , we set inequalities describing a polygon which has n_p vertices equally distributed around \mathbf{h}_0 . The distance between each vertex and the hip projection to the terrain is computed as $\sqrt{l_{max}^2 - h_i^2}$.

D. Support polygon sequence

The method used to generate a sequence of support polygons is similar to the one presented in [2]. Each gait is described by a contact schedule, which in turn defines lift-off and touch-down events for each leg over a complete gait stride. Between each pair of events, a support polygon is defined by the convex hull of the location of the expected contact points, as well as its time duration in seconds. The edges of each support polygon are used to generate the inequality constraints for the ZMP location, such that the optimized motion is dynamically stable.

E. Gait Switching

To fully exploit the capability of executing different gaits, we have developed a gait switching algorithm which enables the locomotion framework to safely and quickly switch from a currently *active gait* to a *desired gait*. The software allows the latter to be selected by the operator.

As described in section III, the motion optimization horizon is computed as the time duration of the active gait. Hence, by changing the contact schedule at the end of the current stride, we avoid jumps in the solution of the optimization. To properly append a new contact schedule to the current one, we first try to connect the active gait with the desired one by selecting lift-off events from the former and combine them with touch-down events from the latter. This can fail for several reasons: there are no overlapping swing phases across active and desired gait at the transition; a new phase event is detected (a foot touch-down or lift-off) that neither corresponds to the active nor the desired gait; or the swing phases would become numerically too large or small at the transition. In case of a failure, the transition is obtained by adding a full stance phase between the gait transitions. Additionally, each gait has its own stride duration which is updated in the vicinity of a transition.

The transition is seen by the optimizer as a change of the incoming sequence of support polygons. This allows the adaption of the body trajectory to the desired gait before the switch occurs.

IV. CONTROL

To track the reference motion plan we use a Whole-Body Controller (WBC) which relies on the Hierarchical Optimization [2] (HO) framework to optimize for the generalized

TABLE I
THE LIST OF PRIORITIZED TASKS USED TO CONTROL THE ROBOT AND TRACK THE MOTION PLANS. PRIORITY 1 IS THE HIGHEST.

Priority	Task
1	Floating base equations of motion
2	Torque limits
3	Friction cone and λ modulation
4	No motion at the contact points
5	Center of mass linear motion
	Center of mass angular motion
	Swing leg motion tracking
	Contact force minimization

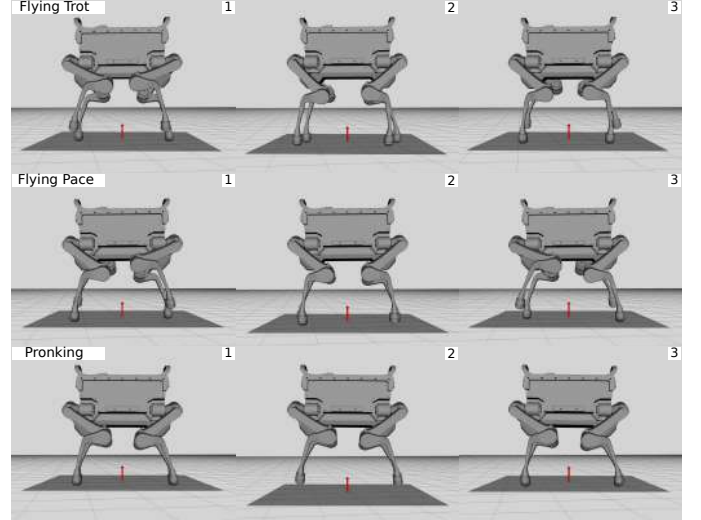


Fig. 5. The motion planner is capable of generating motions for gaits which include full flight phases, such as flying trot, flying pace and pronking.

accelerations $\dot{\mathbf{u}}_d \in \mathbb{R}^{n_u}$ and contact forces $\lambda_d \in \mathbb{R}^{3n_c}$, where n_u is the dimension of the generalized velocity vector \mathbf{u} , and n_c is the number of feet that are in contact with the environment. Table I summarizes the list of tasks, together with their priorities in the control hierarchy, which are used in our controller. Each of these tasks, which specifies equality or inequality constraints on the optimal solution, is solved as a QP problem. From the optimal solution $\dot{\mathbf{u}}^*$ and λ^* , we compute the control torques $\boldsymbol{\tau}$ as

$$\boldsymbol{\tau} = \mathbf{M}_j \dot{\mathbf{u}}^* + \mathbf{h}_j - \mathbf{J}_{s_j}^T \lambda^*, \quad (21)$$

where \mathbf{M}_j , \mathbf{h}_j and \mathbf{J}_{s_j} are, respectively, the rows of the mass matrix, velocity dependent terms and support Jacobian (described in section II) relative to the actuated joints.

V. RESULTS

The framework described in the previous sections was tested in simulation and on the real hardware. A video demonstrating the capabilities of our approach, as well as the experiments discussed in this section, can be found online¹.

¹<https://youtu.be/wQpLzoEzxs8>

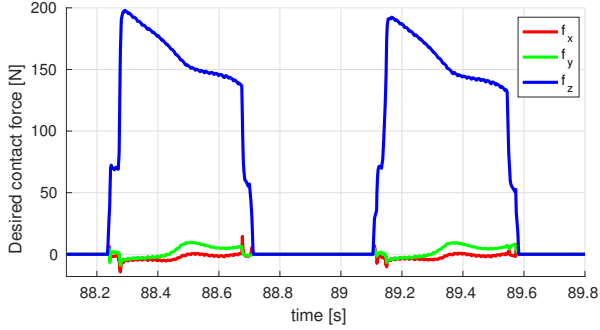


Fig. 6. The desired contact forces for the left hind leg of ANYmal over two strides of a trotting gait computed by the motion controller.

A. Simulation

We have tested our approach by executing the motion plans in the robot simulation toolbox *Gazebo*. Fig.5 depicts snapshots of the execution of a flying trot, a flying pace and a pronking gait, while Fig.6 show the contact forces computed by the whole-body controller while trotting forwards. The optimized motions are tracked by a hierarchical whole-body controller, as discussed in section IV. The user can steer the robot in any desired direction by using an operator device (e.g. a joystick). A graphical user interaction module has been designed to allow to trigger a switch between the available gaits.

B. Experiments

Our experiments were conducted on ANYmal [12], an accurately torque-controllable quadrupedal robot. Control signals are generated in a 400Hz control loop which runs on the robot's on-board computer (Intel i7-7600U, 2.7 - 3.5GHz, dual core 64-bit) together with state estimation [3]. For modeling and computation of kinematics and dynamics, we use the open-source Rigid Body Dynamics Library [15] (RBDL), which is a C++ implementation of the algorithms described in [8]. To solve the nonlinear optimization problem described in section III, we use a custom library which implements the SQP framework, which solves the nonlinear program by iterating through a sequence of Quadratic Programming (QP) problems. Each QP is numerically solved using the QuadProg++ [14] library, an off-the-shelf open source QP solver which implements the Goldfarb-Idnani active-set method [11]. To test the capability of producing full flight phases, we have tested a squat jump (see Fig.8) and a pronking gait (see Fig.9) on ANYmal. To execute the squat jump, the optimizer plans a motion that initially lowers the torso to build up speed and finally jump. The gait switching module has been tested by commanding an online switch between a trot and a dynamic lateral walk which exhibits full swing phase (see Fig.7) overlap between each pair of successive swing legs.

VI. CONCLUSIONS AND FUTURE WORK

We have shown that by extending our previous results by optimizing for the full center of mass position, as well as

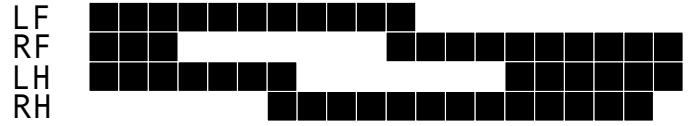


Fig. 7. The gait pattern of a dynamic lateral walk which exhibits swing phase overlap over each pair of successive swing leg. The dark regions represent stance phases.

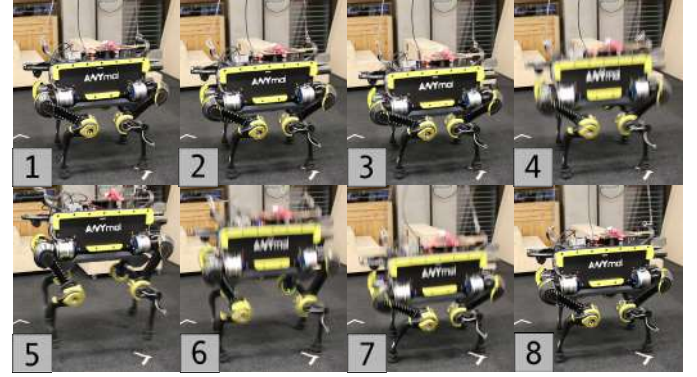


Fig. 8. A sequence demonstrating a squat jump executed on ANYmal.

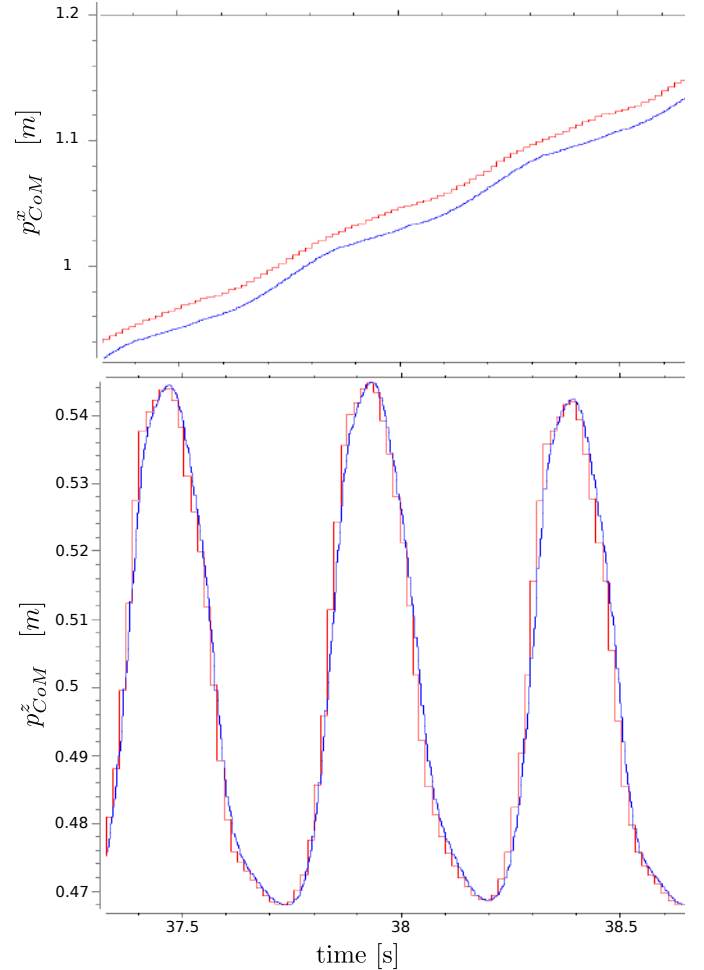


Fig. 9. The x and z components of the center of mass motion generated to execute a pronking gait. The reference motion is in red, while the measured one is depicted in blue.

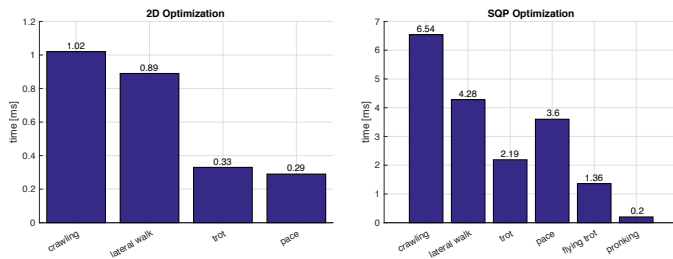


Fig. 10. Comparison of the average computation time needed to optimize for different gaits in our current and previous work. The 2D model used in [2] leads to linear ZMP constraints and the resulting optimization can be handled with a simpler QP solver. The full 3D model accounts for the z component of CoM and, due to nonlinear stability constraints, is solved with a more computationally demanding SQP. This, however, allows to vastly increase the variety of gaits that can be executed.

parallelizing the optimization of footholds, we are now capable of executing a broader range of gaits, including trotting, dynamic lateral walk, jumping in place and pronking. To execute transitions between them, we rely on a dedicated gait switching module. The motion optimization relies on a Sequential Quadratic Programming framework to solve the nonlinear ZMP constraints. Although more complex and more computationally demanding than our previous work, we are still able to produce motion plans online and execute them in an MPC-like fashion. This means that we are able to continuously plan online using the most recent state of the robot while we track the currently available motion plan. Fig.10 summarizes the computation time needed to optimize the torso motion for different gaits. A hierarchical whole-body controller tracks the motion plans and provides disturbance rejection. The natural continuation of this work will be to augment the framework with perception of the environment. This will allow to add additional constraints on the foothold selection. Additionally, this would allow to compute surface normals which would improve the robustness of the optimized motion plans over rough terrain.

REFERENCES

- [1] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter. Perception-less terrain adaptation through whole body control and hierarchical optimization. In *2016 IEEE-RAS 16th Int. Conf. on Humanoid Robots (Humanoids)*, pages 558–564, Nov 2016.
- [2] 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 2017), Vancouver, Canada, September 2428, 2017*. IEEE, 2017.
- [3] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart. State estimation for legged robots on unstable and slippery terrain. In *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 6058–6064. IEEE, nov 2013.
- [4] S. Caron, Q. C. Pham, and Y. Nakamura. Zmp support areas for multicontact mobility under frictional constraints. *IEEE Transactions on Robotics*, 33(1):67–80, Feb 2017.
- [5] J. Engelsberger, P. Kozowski, C. Ott, and A. Albu-Schffer. Biologically inspired deadbeat control for running: From human analysis to humanoid control and back. *IEEE Transactions on Robotics*, 32(4):854–867, Aug 2016.
- [6] J. Engelsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and A. Albu-Schffer. Overview of the torque-controlled humanoid robot toro. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 916–923, Nov 2014.
- [7] P. Fankhauser, C. D. Bellicoso, C. Gehring, R. Dub, A. Gawel, and M. Hutter. Free gait - an architecture for the versatile control of legged robots. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 1052–1058, Nov 2016.
- [8] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer US, Boston, MA, 2008.
- [9] 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)*, pages 5129–5135. IEEE, sep 2015.
- [10] C. Gehring, S. Coros, M. Hutter, 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*, 23(1):34–43, March 2016.
- [11] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27(1):1–33, 1983.
- [12] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, P. Fankhauser, M. Bloesch, R. Diethelm, and S. Bachmann. ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.
- [13] 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*, pages 2665–2670. IEEE, may 2010.
- [14] Luca Di Gaspero. QuadProg++. Available at <http://quadprog.sourceforge.net/>, 1998.
- [15] Martin Felis. Rigid Body Dynamics Library. Available at <http://rbdlib.bitbucket.org/>.
- [16] R. McGhee and A. Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3:331–351, aug 1968.
- [17] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.
- [18] H.-W. Park, P. M. Wensing, and S. Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. *Robotics: Science and Systems*, 2015.
- [19] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(5):630–637, Sept 2004.
- [20] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell. Design of hyq - a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.
- [21] M. Vukobratovic and B. Borovac. Zero-moment point thirty five years of its life. *International Journal of Humanoid Robotics*, 01(01):157–173, 2004.
- [22] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli. Fast trajectory optimization for legged robots using vertex-based zmp constraints. *IEEE Robotics and Automation Letters (RA-L)*, 2:2201–2208, oct 2017.
- [23] A. W. Winkler, C. Mastalli, M. Focchi, D. G. Caldwell, and I. Havoutis. Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain. *IEEE International Conference on Robotics and Automation*, pages 5148–5154, 2015.