Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds

Michael Neunert, Farbod Farshidian, Alexander W. Winkler, Jonas Buchli

***Abstract*—In this work we present a Trajectory Optimization framework for whole-body motion planning through contacts. We demonstrate how the proposed approach can be applied to automatically discover different gaits and dynamic motions on a quadruped robot. In contrast to most previous methods, we do not pre-specify contact-switches, -timings, -points or gait patterns, but they are a direct outcome of the optimization. Furthermore, we optimize over the entire dynamics of the robot, which enables the optimizer to fully leverage the capabilities of the robot. To illustrate the spectrum of achievable motions, we show eight different tasks, which would require very different control structures when solved with state-of-the-art methods. Using our Trajectory Optimization approach, we are solving each task with a simple, high level cost function and without any changes in the control structure. Furthermore, we fully integrate our approach with the robot's control and estimation framework such that we are able to run the optimization online. Through several hardware experiments we show that the optimized trajectories and control inputs can be directly applied to physical systems.**

***Index Terms*—Multilegged Robots, Motion and Path Planning, Optimization and Optimal Control**

I. Introduction

In motion planning and control, one major challenge is to specify a high level task for a robot without specifying how to solve this task. In legged locomotion such tasks include reaching a goal position or manipulating an object without specifying gaits, contacts, balancing or other behaviors. Trajectory Optimization (TO) recently gained a lot of attention in robotics research since it promises to tackle some of these problems. Ideally, it would solve complex motion planning tasks for robots with many degrees of freedom, leveraging the full dynamics of the system. However, there are two challenges of TO. Firstly, TO problems are hard problems to solve, especially for robots with many degrees of freedom and for robots that make or break contact. Therefore, many approaches add heuristics or pre-specify contact points or sequences. However, this then defines again how the robot is supposed to solve the task, affecting optimality and generality. Secondly, TO cannot be blindly applied to hardware but requires an accurate model as well as a good control and estimation framework. In this work, we are addressing both issues. In our TO framework, we only specify high level tasks, allowing the solver to find the optimal solution to the problem, optimizing over the entire dynamics and automatically discovering the contact sequences and timings. Second, we also demonstrate how such a dynamic task can be generated online. This work does not present a general tracking controller suitable for all trajectories. However, we show a successful integration of TO with our estimation and control framework, allowing the execution of several tasks on hardware even under disturbances.

Manuscript received: September, 10, 2016; Revised December, 12, 2017; Accepted January, 11, 2017.

This paper was recommended for publication by Nikos Tsagarakis upon evaluation of the Associate Editor and Reviewers' comments. *This research has been funded through a Swiss National Science Foundation Professorship award to Jonas Buchli and the NCCR Robotics.

All authors are with the Agile & Dexterous Robotics Lab, ETH Zürich, Switzerland. {neunertm, farbodf, winklera, buchlij}@ethz.ch

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

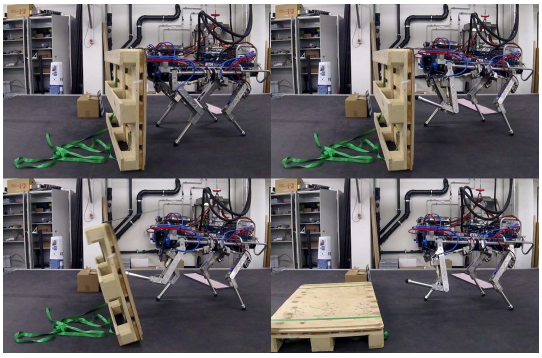


Fig. 1. Sequence of images during execution of the rough manipulation task in hardware. Time progresses line-wise from top left to bottom right.

TO tries to solve a general, time-varying, non-linear optimal control problem. There are various forms of defining and tackling such a control problem. An overview can be found in [1]. With the increase of computational power, TO can be applied to higher dimensional systems like legged robots. Thus, it gained a lot of attention in recent years and impressive results were demonstrated [2], [3], [4]. Yet, these approaches do not present hardware results and [2], [3] do not discuss how to stabilize the trajectory. One of the conceptually closest related work is [5]. However, no gait discovery or very dynamic motions are shown and hardware results are missing. Later work [6] includes hardware results but the planning horizon is short, the motions are quasi-static and contact changes are slow. In [7] TO through contacts is demonstrated on hardware. While the results are promising, the approach is only tested on a single leg platform and for very simple tasks. In general, work that demonstrates TO through contacts applied to physical, legged systems is rare [8], especially on dynamic motions and torque controlled robots. TO with dynamic motions demonstrated on quadruped hardware is presented in [9]. The results of this work are convincing and the authors are also considering actuator dynamics. However, their approach differs in key areas to the presented work: Contact sequences are pre-specified and their approach optimizes over control parameters for a fixed control structure, rather than whole body trajectories and control inputs. Additionally, they are using black box optimization. TO is also used for kinematic [10] or kinodynamic planning [11]. However, these approaches fall

short for evaluating stability or generating dynamic motions such as jumping. Also, not all kinematic plans are dynamically feasible since they may exceed torque limits or friction cones. Often motion planning through contacts is solved using a hierarchy of optimization problems [12], [13], [14], [7]. The issue is that the lower hierarchies need to respect the constraints of the higher hierarchies, often leading to heuristics which impair the optimality of the overall solution.

A. Contributions

In this work, we apply TO in form of Sequential Linear-Quadratic (SLQ) control [15] to the quadruped robot HyQ [16], both in simulation and on hardware. Our work is one of very few examples, where TO is used to plan dynamic motions through contact changes on a whole-body 3D model and where these trajectories are shown to work on hardware. Compared to previous work on TO through contacts [2], [3], [4], [10], these hardware results help to understand the capabilities and limitations of the approach on physical systems. Furthermore, this is possibly the first time that a gait-free whole-body TO approach is shown on quadruped hardware. In contrast to state-of-the-art approaches on quadrupeds [9], neither contact switching times nor contact events nor contact points are defined a priory. Instead they are a direct outcome of the optimization. This allows us to generate a diverse set of motions and gaits using a single approach. By using an efficient formulation based on Differential Dynamic Programming and our high-performance solver, optimization times for these tasks usually do not exceed a minute, even for complex trajectories, outperforming state-of-the-art approaches [2], [9]. Thus, the TO can be run online and adapt to the given situation. Hence, this work is also one of the earliest examples of whole body TO run alongside a control and estimation pipeline for a legged robotic system. Many TO approaches [2], [3], [10] simply "play-back" trajectories for visualization rather than testing them in a control framework. However, only by executing trajectories in the loop with estimators and controllers, we can see physical defects, argue about stability and verify our model assumptions.

II. Trajectory Optimization

A. Optimal Control Problem

In this work, we consider a general non-linear system

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{1}$$

where $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ denote state and input trajectories respectively. Our TO approach tries to find the optimal state and control trajectories by solving a finite-horizon optimal control problem which minimizes a given cost function

$$J(\boldsymbol{x}, \boldsymbol{u}) = h\left(\boldsymbol{x}(t_f)\right) + \int_{t=0}^{t_f} l\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) dt \tag{2}$$

To solve this optimal control problem, we apply Sequential Linear Quadratic Control [15], which optimizes a linear, time-varying feedback and feedforward controller of the form $\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}_{ff}(t) + \mathcal{K}(t)\boldsymbol{x}(t)$ where $\mathcal{K}(t)$ is the time-varying control gain and $u_{ff}(t)$ the feedforward control action.

Algorithm 1: SLQ Algorithm

Input: System dynamics: $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)$
Input: Cost function: $J = h\left(\boldsymbol{x}(t_f)\right) + \int_{t=0}^{t_f} l\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)$
Input: Initial state and control law: $\boldsymbol{x}(0), \boldsymbol{u}(\boldsymbol{x}, t)$
repeat
- $\boldsymbol{x}(0...t_f) \leftarrow \int_0^{t_f} f(\boldsymbol{x}(t), \boldsymbol{u}(\boldsymbol{x}, t))$ `// Rollout dynamics`
- $\boldsymbol{u}_a(0...t_f) \leftarrow \boldsymbol{u}(\boldsymbol{x}(0...t), 0...t)$ `// Record controller`
- **for** $t = 0$ **to** $t_f - 1$ **do**
  - `Linearize the system dynamics`
    $\boldsymbol{A}(t) \leftarrow \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}|_{\boldsymbol{x}(t), \boldsymbol{u}(t)}$
    $\boldsymbol{B}(t) \leftarrow \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}|_{\boldsymbol{x}(t), \boldsymbol{u}(t)}$
  - `Quadratize cost function`
    $q(t) \leftarrow J(t)$, $\boldsymbol{q}(t) \leftarrow \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{x}}|_{\boldsymbol{x}(t), \boldsymbol{u}(t)}$, $\boldsymbol{Q}(t) \leftarrow \frac{\partial^2 \boldsymbol{J}}{\partial \boldsymbol{x}^2}|_{\boldsymbol{x}(t), \boldsymbol{u}(t)}$
    $\boldsymbol{P}(t) \leftarrow \frac{\partial^2 \boldsymbol{J}}{\partial \boldsymbol{x} \partial \boldsymbol{u}}|_{\boldsymbol{x}(t), \boldsymbol{u}(t)}$
    $\boldsymbol{r}(t) \leftarrow \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{u}}|_{\boldsymbol{x}(t), \boldsymbol{u}(t)}$, $\boldsymbol{R}(t) \leftarrow \frac{\partial^2 \boldsymbol{J}}{\partial \boldsymbol{u}^2}|_{\boldsymbol{x}(t), \boldsymbol{u}(t)}$
- **for** $t = t_f - 1$ **to** 0 **do**
  - `Solve the Riccati-like difference equations:`
    $\boldsymbol{P}(t) \leftarrow \boldsymbol{Q}(t) + \boldsymbol{A}^T(t)\boldsymbol{P}(t+1)\boldsymbol{A}(t) + \boldsymbol{K}^T(t)\boldsymbol{H}\boldsymbol{K}(t) + \boldsymbol{K}^T(t)\boldsymbol{G} + \boldsymbol{G}^T\boldsymbol{K}(t)$
    $\boldsymbol{p}(t) \leftarrow \boldsymbol{q} + \boldsymbol{A}^T(t)\boldsymbol{p}(t+1) + \boldsymbol{K}^T(t)\boldsymbol{H}\boldsymbol{l}(t) + \boldsymbol{K}^T(t)\boldsymbol{g} + \boldsymbol{G}^T\boldsymbol{l}(t)$
    $\boldsymbol{H} \leftarrow \boldsymbol{R}(t) + \boldsymbol{B}^T(t)\boldsymbol{P}(t+1)\boldsymbol{B}(t)$
    $\boldsymbol{G} \leftarrow \boldsymbol{B}^T(t)\boldsymbol{P}(t+1)\boldsymbol{A}(t)$
    $\boldsymbol{g} \leftarrow \boldsymbol{r}(t) + \boldsymbol{B}^T(t)\boldsymbol{p}(t+1)$
    $\boldsymbol{K}(t) \leftarrow -\boldsymbol{H}^{-1}\boldsymbol{G}$ `//feedback update`
    $\boldsymbol{l}(t) \leftarrow -\boldsymbol{H}^{-1}\boldsymbol{g}$ `//feedforward increment`
- $\alpha \leftarrow 1$ `// Initialize line search`
- **repeat**
  - **Line search**
  - 1. Update the control: $\boldsymbol{u}(\boldsymbol{x}, t) \leftarrow \boldsymbol{u}_a(t) + \alpha\boldsymbol{l}(t) + \boldsymbol{K}(t)\boldsymbol{x}(t)$
  - 2. Forward simulate the system dynamics: $\boldsymbol{x}(0...t_f) \leftarrow \int_0^{t_f} f(\boldsymbol{x}(t), \boldsymbol{u}(\boldsymbol{x}, t))$
  - 3. Compute new cost: $J = h\left(\boldsymbol{x}(t_f)\right) + \int_{t=0}^{t_f} l\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) dt$
  - 4. decrease α by a constant α_d: $\alpha = \alpha/\alpha_d$
- **until** *found lower cost or number of maximum line search steps reached*

until *maximum number of iterations or converged* ($\boldsymbol{l}(t) < l_t$)

B. Sequential Linear Quadratic Control

Sequential Linear Quadratic Control is an iterative optimal-control algorithm. SLQ first rolls out the system dynamics. Then, the non-linear system dynamics are linearized around the trajectory and a quadratic approximation of the cost function is computed. The resulting Linear-Quadratic Optimal Control problem is solved backwards. Since the solution to the Linear-Quadratic Optimal Control problem can be computed in closed form with Ricatti-like equations, SLQ is very efficient. Algorithm 1 summarizes the algorithm. SLQ computes both, a feedforward control action as well as a time-varying linear-quadratic feedback controller. Handling input constraints in SLQ is possible and we do so by saturating the control input [17]. So far SLQ type algorithms were known to be unable to handle state or state-input constraints. However, recent research conducted after the herein presented work, shows that SLQ is able to handle state and state-input constraints without breaking its linear time complexity by using soft-constraints [18] or by modifying the algorithm [19]. While not yet used in this work, these approaches extend the applicability of SLQ.

C. Cost Function

While SLQ can handle non-quadratic cost functions, pure quadratic cost functions increase convergence and are often sufficient to describe complex tasks. Therefore, we assume

our cost function to be of quadratic form

$$J = \bar{\boldsymbol{x}}(t_f)^T \boldsymbol{H} \bar{\boldsymbol{x}}(t_f) + \int_{t=0}^{t_f} \bar{\boldsymbol{x}}(t)^T \boldsymbol{Q} \bar{\boldsymbol{x}}(t) + \bar{\boldsymbol{u}}(t)^T \boldsymbol{R} \bar{\boldsymbol{u}}(t) \; dt \tag{3}$$

where $\bar{x}(t)$ and $\bar{u}(t)$ represent deviations of state and input from a desired state and desired input respectively. These references are fixed, time-invariant setpoints, i.e. there are no state or input trajectories. $\boldsymbol{H}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the weightings for final and intermediate state as well as control input cost respectively. In some instances, we add intermediate state waypoints to the cost function. These temporal costs are weighted by the waypoint cost matrix $\boldsymbol{W}(x,t)$ and defined as

$$W(\boldsymbol{x}, t) = \sum_{n=0}^{N} \hat{\boldsymbol{x}}_n(t)^T \boldsymbol{W}_n \hat{\boldsymbol{x}}_n(t) \sqrt{\frac{\rho_n}{2\pi}} \exp\left(-\frac{\rho_n}{2}(t - t_n)^2\right) \tag{4}$$

where ρ_p defines the "temporal spread" of the cost and t_n defines the time at which the waypoint is active. The waypoint penalizes the deviation from a fixed, time-invariant, desired state $\boldsymbol{x}_{n,d}$ via the difference $\hat{\boldsymbol{x}}_n(t) = \boldsymbol{x}(t) - \boldsymbol{x}_{n,d}$. $\boldsymbol{W}_n$ defines the weighting, which is a purely diagonal matrix. We limit the waypoint usage to two explicit cases: Firstly, when a task cannot possibly be described by a pure quadratic cost function and secondly to increase robustness during hardware experiments. All quadruped gaits shown in Section V are discovered without the use of waypoints.

III. System Model and Robot Description

For the experiments in this paper, we use the hydraulically actuated, quadrupedal robot HyQ [16]. Each of the four legs on HyQ has three degrees of freedom, namely hip abduction/adduction (HAA), hip flexion/extension (HFE) and knee flexion/extension (KFE). Each joint is driven by a hydraulic actuator. Joint torques and positions are measured via load cells and encoders respectively. A hydraulic force control loop is closed at joint level [20].

A. Rigid Body Dynamics

While it is a simplification, torque tracking performance on HyQ is sufficient for modeling the robot as a perfectly torque controlled system. Thus, we assume HyQ to behave like a rigid body system defined as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{J}_c^T \lambda(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{S}^T \tau \tag{5}$$

where $\boldsymbol{M}$ denotes the inertia matrix, $\boldsymbol{C}$ the centripetal and Coriolis forces and $\boldsymbol{G}$ gravity terms. $\boldsymbol{q}$ is the state vector containing the 6 DoF base state as well as joint positions and velocities. External and contact forces λ act on the system via the contact Jacobian $\boldsymbol{J}_c$. Torques created by the actuation system τ are mapped to the actuated joints via the selection matrix $\boldsymbol{S}$. To formulate our dynamics according to Equation (1), we define our state as

$$\boldsymbol{x} = [{}_{\mathcal{W}}\boldsymbol{q} \; {}_{\mathcal{L}}\dot{\boldsymbol{q}}]^T = [{}_{\mathcal{W}}\boldsymbol{q}_B \; {}_{\mathcal{W}}\boldsymbol{x}_B \; \boldsymbol{q}_J \; {}_{\mathcal{L}}\omega_B \; {}_{\mathcal{L}}\dot{\boldsymbol{x}}_B \; \dot{\boldsymbol{q}}_J]^T \tag{6}$$

where ${}_{\mathcal{W}}\boldsymbol{q}_B$ and ${}_{\mathcal{W}}\boldsymbol{x}_B$ define base orientation and position respectively, which are expressed in a global inertial "world" frame $\mathcal{W}$. The base orientation is expressed in Euler angles (roll-pitch-yaw). Base angular and linear velocity are denoted as ω_B and $\dot{\boldsymbol{x}}_B$ respectively and both quantities are expressed in a local body frame $\mathcal{L}$. Joint angles and velocities are expressed as $\boldsymbol{q}_J$ and $\dot{\boldsymbol{q}}_J$ respectively. Expressing base pose and twist in different frames allows for more intuitive tuning of the cost function weights. Using Equations (5) and (6) our system dynamics in Equation (1) become

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \begin{bmatrix} {}_{\mathcal{W}}\dot{\boldsymbol{q}} \\ {}_{\mathcal{L}}\ddot{\boldsymbol{q}} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{R}_{\mathcal{WL}} \; {}_{\mathcal{L}}\dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{S}^T\tau + \boldsymbol{J}_c\lambda(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q})) \end{bmatrix} \end{aligned} \tag{7}$$

where $\boldsymbol{R}_{\mathcal{WL}}$ defines the rotation between the local body frame $\mathcal{L}$ and the inertial "world" frame $\mathcal{W}$. While dropped for readability, $\boldsymbol{R}_{\mathcal{WL}}$ is a function of ${}_{\mathcal{W}}\boldsymbol{q}$ which needs to be considered during linearization. For SLQ we need to linearize the system given in Equation (7). For the derivatives of the upper row with respect to the state $\boldsymbol{x}$ as well as all derivatives with respect to τ we compute analytical derivatives. For the derivatives of the lower row in Equation (7) with respect to the state $\boldsymbol{x}$, we use numerical differentiation.

B. Contact Model

Choosing or designing an appropriate contact model is critical for the performance of TO. For TO it is beneficial to use a smooth contact model which provides good gradients of the dynamics. Yet, a soft model can lead to unphysical effects such as ground penetration or sliding contacts. As a trade-off we are using a non-linear spring-damper contact model extending the model proposed in [15]. We consider two contact models: one collinear and one orthogonal to the surface normal. The orthogonal contact model is defined as

$$\lambda_N = \begin{cases} 0 & p_n \leq 0 \\ (k_n + d_n \dot{p}_n)\frac{p_n^2}{2\alpha_c}\boldsymbol{n}_s & 0 < p_n < \alpha_c \\ (k_n + d_n \dot{p}_n)(p_n - \frac{\alpha_c}{2})\boldsymbol{n}_s & p_n \geq \alpha_c \end{cases} \tag{8}$$

while the tangential model is defined as

$$\lambda_t = \begin{cases} 0 & p_n \leq 0 \\ (k_t p_t \boldsymbol{n}_d + d_t \dot{p}_t \boldsymbol{n}_v)\frac{p_n^2}{2\alpha_c} & 0 < p_n < \alpha_c \\ (k_t p_t \boldsymbol{n}_d + d_n \dot{p}_n \boldsymbol{n}_v)(p_n - \frac{\alpha_c}{2}) & p_n \geq \alpha_c \end{cases} \tag{9}$$

Both models include proportional and derivative terms which can be interpreted as springs with stiffnesses k_n, k_t and dampers with damping ratios d_n, d_t respectively. The range of parameter values used for the contact model are given in Table I. For the normal direction the spring displacement p_n is defined as the ground penetration along the surface normal $\boldsymbol{n}_s$. In tangential direction, the offset p_t is computed between the current contact location and the location where the contact has been established. In case of the tangential model, the force vector λ_t is composed of the spring force in tangential displacement direction $\boldsymbol{n}_d$ and a damping force in displacement velocity direction $\boldsymbol{n}_v$. Both, the normal and

TABLE I
TYPICAL VALUES FOR CONTACT MODEL PARAMETERS

α_c	k_n	d_n	k_t	d_t
0.01	8000...90000	2000...50000	0...5000000	2000...5000

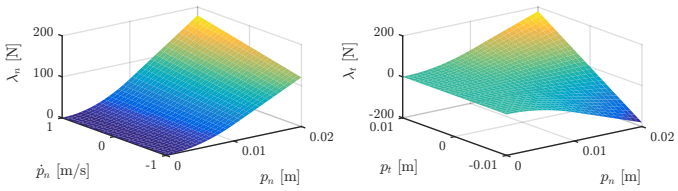


Fig. 2. Visualization of the contact model. The left plot shows the contact force in surface normal direction as a function of penetration and its derivative. The right plot shows the force in tangential direction as a function of penetration and tangential displacement.

tangential contact models are visualized in Figure 2. To avoid discontinuities during contact changes, the models are non-linear towards zero ground penetration, i.e. for $p_n < \alpha$, where α_c is a smoothing coefficient.

Friction and friction limits are considered by the contact model via friction cones. Thus, they are not included as constraints but are a part of the dynamical system. This allows the TO algorithm to reason about possible slippage and contact force saturation. In this work, we assume that static and dynamic friction coefficients are the same. Therefore, the friction cone can be expressed as a contact force saturation $\boldsymbol{F}_{t,sat} = max(||\boldsymbol{F}_t||, \mu F_n)\boldsymbol{n}_t$, where $\boldsymbol{F}_t$ and F_n are contact forces parallel and orthogonal to the surface. The normal vector $\boldsymbol{n}_t$ defines the direction of the (unsaturated) tangential contact force $\boldsymbol{F}_t$. The friction coefficient is denoted by μ. Saturating the contact forces results in discontinuities in the overall dynamics. However, due to numerical differentiation, gradient information is recovered and line search helps to mitigate the issue in practice. One could consider replacing the hard limit by a (conservative) soft limit. While the presented contact model works well for obtaining trajectories, there are limitations when it comes to the robustness of the obtained trajectories. While the friction cone is considered, violations of it, i.e. sliding contacts, are not penalized. However, such slippage can cause issues during execution. Additionally, the previously mentioned trade-off between softness for gradients and physical accuracy remains. These and other limitations are discussed in Subsection VI-B.

IV. STATE ESTIMATION AND TRACKING CONTROL

Our TO is fully integrated into our estimation and control framework, shown in Figure 3. This framework consists of base state and ground estimators, the SLQ solver and tracking controllers. The base estimator and controller run at 250 Hz, while the joint controller operates at 1 kHz.

A. State Estimation

SLQ assumes full state-feedback. We directly measure joint positions using encoders and compute joint velocities using numerical differentiation. These measurements are fused with IMU data to obtain a base state estimate [21]. While in

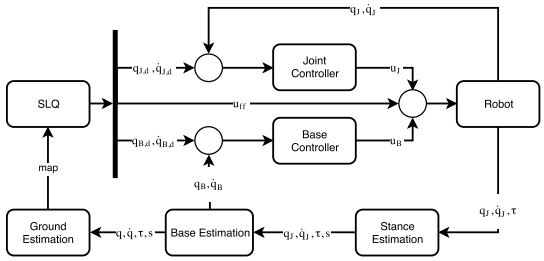


Fig. 3. Overview of the control and estimation pipeline. Base, stance and ground estimators provide information about the location and contact configuration of the robot. Joint and base controllers stabilize the robot.

the TO problem, we assume a soft contact model, our base controller uses discrete contact state information. We obtain the contact state by estimating ground reaction forces from joint torques and compare them to a fixed threshold. We then fit a ground plane to the last contact points of all feet, which provides an estimate of the elevation and contact surface normal. This plane is then also the reference for the soft contact model in the TO. While this is a shortcut to ground estimation or elevation mapping, the presented approach is not limited to co-planar contacts. However, to maintain good gradient information the ground height and normal should be continuous and differentiable.

B. Tracking Controller

SLQ does not only optimize feed-forward control action but also a feedback controller. While we have successfully applied theses feedback gains to robotic hardware [22], we are not using the optimized feedback gains in this work. The gains are computed for a linearized model and thus significantly depend on the linearization point. During execution, the robot's state will deviate from this nominal point and can thus lie outside of the region of attraction of the controller. Therefore, we use a combination of a joint and base state controller instead. A base controller allows us to directly track the base state and tune feedback gains on these states intuitively. Yet, for swing legs, we still require a joint controller. Additionally, a joint controller on stance legs increases robustness. Hence, we use PD controllers on both, the base and all joint states. The base PD task space controller can be described as

$$\boldsymbol{F}_{cog} = \boldsymbol{P}_x({}_{\mathcal{L}}\boldsymbol{x}^*_{base} - {}_{\mathcal{L}}\boldsymbol{x}_{base}) + \boldsymbol{D}_x({}_{\mathcal{L}}\dot{\boldsymbol{x}}^*_{base} - {}_{\mathcal{L}}\dot{\boldsymbol{x}}_{base}) \quad (10)$$

which regulates errors between desired $({}_{\mathcal{L}}\boldsymbol{x}^*_{base}, {}_{\mathcal{L}}\dot{\boldsymbol{x}}^*_{base})$ and actual $({}_{\mathcal{L}}\boldsymbol{x}_{base}, {}_{\mathcal{L}}\dot{\boldsymbol{x}}_{base})$ base state. The desired body wrench $\boldsymbol{F}_{cog}$ is applied to the robot by converting it to forces at the feet $\boldsymbol{\lambda}_c$ and then mapping them to the joint torques through $\tau_{fb} = \mathbf{J}_c^T \lambda_c$. These torques are then added to the feedforward control action obtained from SLQ. Since it is a model based approach. the SLQ control action already includes torques that counteract gravity and thus, gravity compensation does not need to be added in Equation (10).

V. EXPERIMENTS

To validate the approach, we describe different choices of cost functions and show the resulting gaits and motions. For

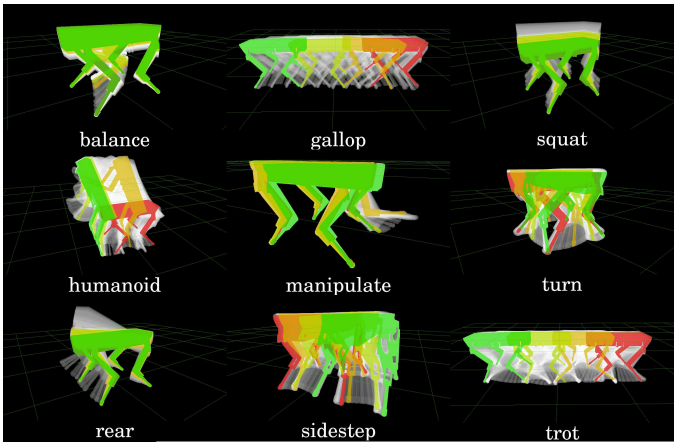
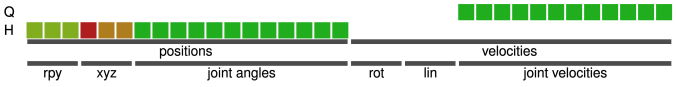


Fig. 4. Time series of the optimized trajectories for each task. Poses are shown as a color gradient over time ranging from red (initial pose) to green (final pose). Intermediate poses are indicated in transparent white. All displayed motions contain contact switches during dynamic maneuvers. These contact switches result from optimization and are not pre-specified.

all tasks executed on hardware we use hard input constraints. Not all tasks are executable on hardware for which a thorough discussion is provided in Subsection VI-B.

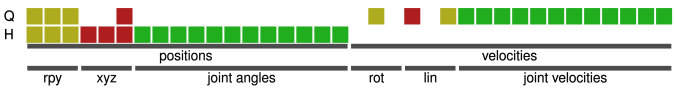
For each task, the cost function is shown as a color code below the heading. The colors indicate the individual, relative weightings of the diagonal entries of each weighting matrix, ranging from lowest (green) to highest (red) on a logarithmic scale. No color means that the respective value is zero and all off-diagonal elements are zero as well. The input cost matrix $\boldsymbol{R}$ is set to identity in all experiments. All tasks are initialized with a simple stance controller and no contact sequence or timings are given. Snapshots of all tasks are shown in Figure 4. The optimized trajectory and hardware experiments are shown in the video. All cost functions, solver parameters and the robot model are provided as supplementary material[1].

A. Galloping



The first gait we demonstrate is galloping. The galloping gait is a direct outcome of setting the final cost terms to penalize the deviation from a desired final pose which is 2 m in front of the robot. Additionally, we add some regularization on the base and leg motion to prevent excessive motions of the body or the limbs. As the results in Figure 5 illustrates, we obtain a gallop motion with 9 steps which includes acceleration and deceleration. Finally, the robot reaches a desired position at $x = 2.0$ m. As expected, we see significant pitch motion of the upper body. From Figure 6 we can tell that mostly the hind legs are used for acceleration.

B. Trotting



[1]https://bitbucket.org/neunertm/hyq_slq_config

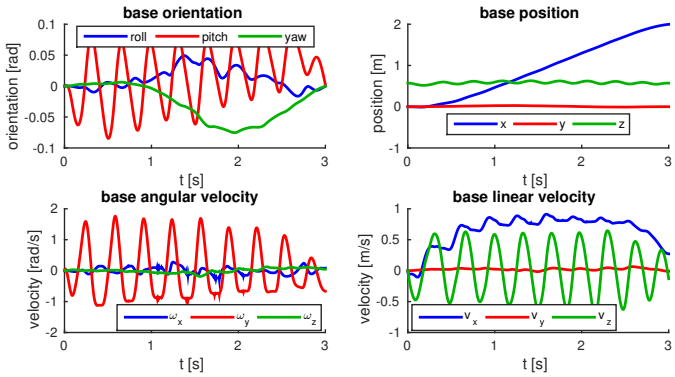


Fig. 5. Plots of base pose and base twist during galloping as optimized by SLQ. The robot takes in total 9 galloping steps. The desired final position at $x = 2.0$ m is reached with good accuracy.

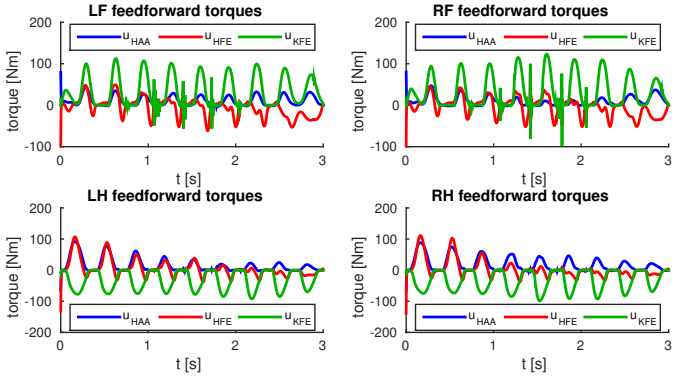
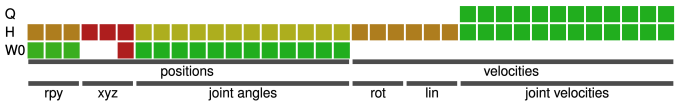


Fig. 6. Torque profiles of the different joints during a galloping motion. As the lower plots show, the hind legs and especially the HFE joints, contribute greatly to the acceleration. The front leg torques seem to contribute fairly evenly to the galloping motion throughout the trajectory.

One hypothesis for the previous task resulting in a galloping behavior is the short time horizon and no penalty on the orientation. If we increase the time horizon to 8 s and penalize the base motion, i.e. we give HyQ more time and encourage smoother base motions, we see that the optimization prefers trotting over galloping. The trot consists of four steps per diagonal leg pair with almost constant stride length. By setting the desired position to the side instead of to the front, the resulting trajectory is a sidestepping motion. If only a desired yaw angle is set, the robot turns on the spot. In all cases the diagonal leg pairs are moved together.

C. Squat Jump



Next, we test if our TO approach can leverage and reason about the dynamics of our system. We do so by adding an intermediate waypoint cost term for the base pose at 0.2 m above the initial base height. Since reaching there by extending the legs exceeds the kinematic limits of the robot, a jump is required. By adding a penalty on the deviation from default joint positions to the waypoint, we encourage a larger ground clearance at the apex. After running our optimization, we

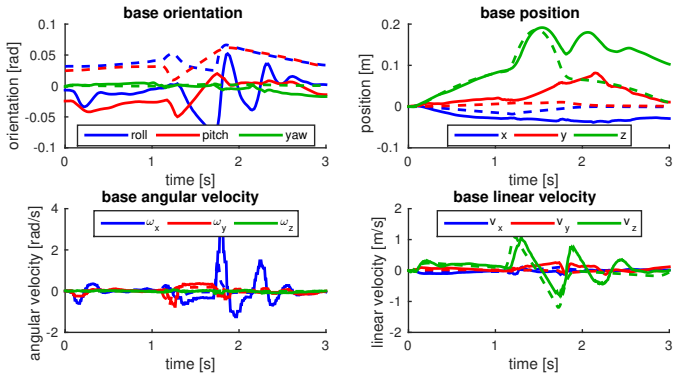


Fig. 7. Plots of optimized (dashed) and executed (solid) base state during a squat jump on hardware. The robot reaches the desired/planned apex height. Due to insufficient damping, there is a small rebound motion after landing.

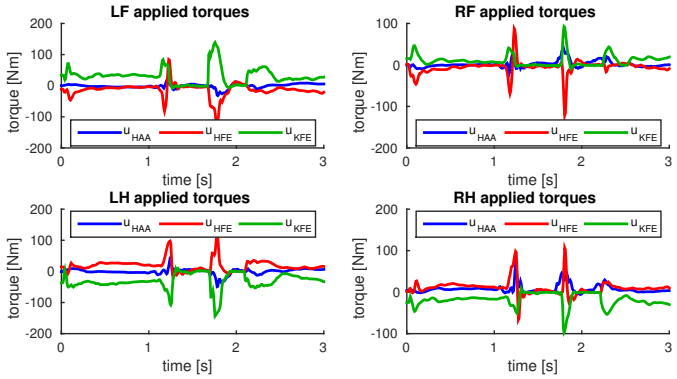
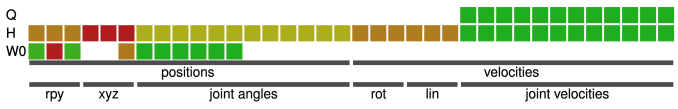


Fig. 8. Plots of the applied torques during a squat jump on hardware. There are two distinct torque spikes produced during take-off and landing.

obtain a near symmetric squat jump. The overall optimization spans the entire motion, e.g. preparation for lift-off from default pose, the lift-off itself, going to default pose in the air, landing and returning to the default pose. The apex waypoint is localized in time but contact switches and timings are optimized. Figure 7 shows the optimized and executed squat jump trajectory. Throughout the task, the base stays level. The desired apex height of 0.2 m is reached with millimeter accuracy (measured 0.192 m). Torques are shown in Figure 8 which also illustrate lift-off and touch-down times. In the left image of Figure 9, we can further see that the robot's legs are in their default position in mid-air as specified in the cost function.

D. Rearing



For the next task, we are using a similar cost function as for the squat jump. Instead of penalizing the deviation from a neutral base pose, we penalize deviations from a 30° pitched base orientation and we lower the desired apex height to 0.7 m. The final trajectory is a rearing motion, where HyQ lifts off with the front legs, reaches the apex position and finally returns to full contact as well as its default pose.

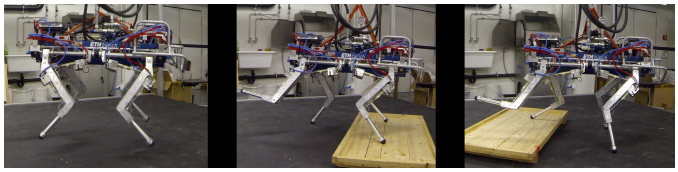
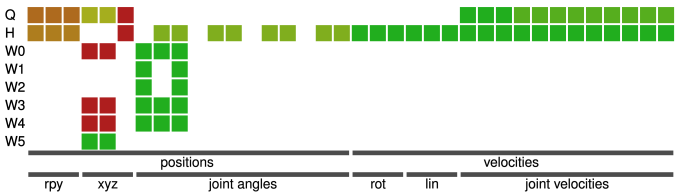


Fig. 9. Hardware tests for the squat jump (left) and the manipulation under disturbance (center and right). As the algorithm is run online, the motion is optimized to the terrain, e.g. leading to different hind leg joint angles.

E. Diagonal Balance

In order to demonstrate that SLQ can also find statically unstable trajectories, we are demonstrating a diagonal balance task. Here, we use a waypoint term in our cost function again. This term penalizes the orientation and height of the base. Furthermore, it encourages the robot to pull up its legs by bending HFE and KFE of the left front and right hind leg. The final trajectory shows the expected balancing behavior. Again a screenshot at apex is shown in Figure 4 and the videoshows the full motion. Interestingly, while we are using a single intermediate term with a single time point and absolutely symmetric costs, the lift-off and touch-down of the swing legs is not synchronous but the front left leg lifts-off later and touches down earlier. This asymmetry most likely stems from the asymmetric location of the Center of Gravity and asymmetric inertia of the robot's main body.

F. Manipulate



As a last task we test a "rough" manipulation motion where HyQ pushes over an obstacle with its front left leg. This task involves submotions such as shifting the robot's CoG in the support polygon of the three stance legs, executing the push motion and shifting the CoG back. While in classic approaches these motions would possibly be all hand coded, they directly result from a single cost function in our TO approach. While the resulting trajectory works perfectly fine in simulation, it lacks robustness on hardware. Since our TO approach is deterministic and we penalize control input, the algorithm tries to minimize the shift of the CoG, leading to a risky trajectory. For visual purposes only, we add additional intermediate waypoints for the front left leg. While we could easily add the pallet push contact to our optimization, we leave it unmodelled on purpose such that it becomes a disturbance to our controller verifying its robustness. Figure 4 shows a sequence of images of the optimized motion while Figure 1 shows a sequence of images taken during execution. One of the advantages of TO and running it online is its capabilities

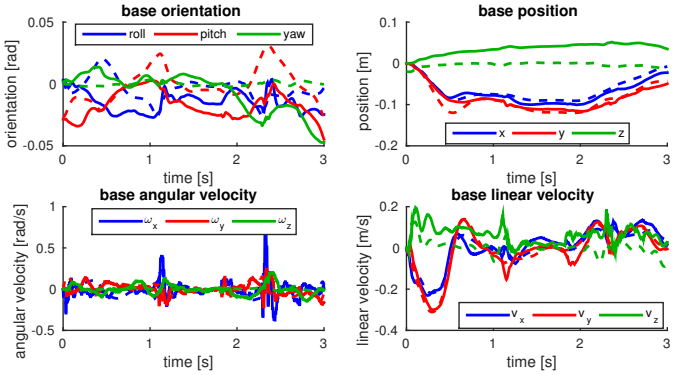
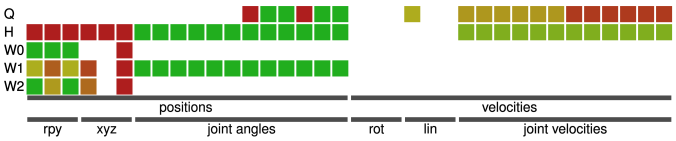


Fig. 10. Base pose and twist trajectories (solid) and their respective references (dashed) for the rough manipulating task on hardware. The plots show that the planned and executed base trajectories only slightly deviate.

to adjust to different initial conditions. We vary the initial conditions by placing HyQ's feet over a wooden platform in different configurations as shown in Figure 9. All these tests are included in the video.In all motions, CoG shift and leg lift-off overlap in time, i.e. the result is a fluid, dynamic motion. Independent of the initial inclination, the robot levels its base pose during execution. Also, depending on the initial configuration, lift-off timings of the front left leg vary significantly to obtain an optimal motion. This underlines the importance of automatically discovering contact timings. For the experiment in Figure 1, the base pose/twist tracking is shown in Figure 10.

G. Humanoid Walk



While exceeding the capabilities of our hardware, we evaluate a humanoid walking task where HyQ gets up on its hind legs, moves to a target point in front and then balances there. In contrast to a humanoid with ankles, HyQ only has point feet, increasing under-actuation and the difficulty of the task. The first cost function waypoint, widely spread in time, penalizes deviations in base orientation and height. This ensures that HyQ stays upright during the entire task after getting up. The stand up motion is encouraged by the second waypoint penalizing base orientation, height and changes in forward position. We add a third waypoint one second before the end of the time horizon, defining the target pose and orientation. While the last waypoint and the final cost specify the same base pose, we separate them in time to demonstrate that HyQ can stay upright and stabilize in place for a short time. Before getting up HyQ pulls its hind left leg in, moving the contact point closer below the center of gravity. Also, it uses the front left leg ("left arm") to get up, resulting in a very natural, coordinated, asymmetric motion. After getting into a two-leg standing phase, forward motion is initiated by a short symmetric hopping but quickly changes to a walking pattern. Such non-trivial motions are hard to obtain from fixed timing methods [18].

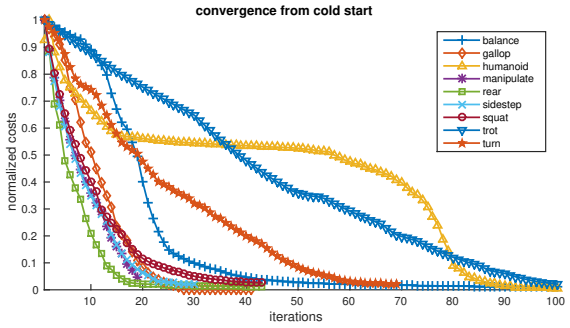


Fig. 11. Convergence rates for different tasks, which seem to be influenced by the length and complexity of each task. The fastest convergence is observed in the rearing task. Trotting converges the slowest.

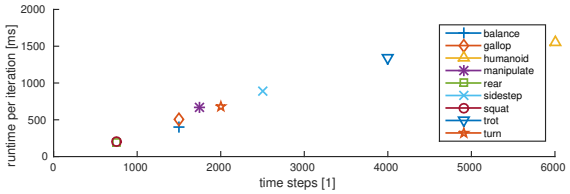


Fig. 12. Runtime per iteration for different tasks. From a theoretical point of view, the relation between number of time steps and the runtime per iteration should be linear. This plot nicely supports this hypothesis.

VI. Discussion

A. Runtime and Convergence

When running TO online, runtime and convergence become a major concern since these measures define how long the robot "thinks" before executing a task. Especially in dynamic environments, we want to keep the optimization procedures short. In this section, we will look at both the number of iterations for each task as well as the runtime of each iteration. This gives us an indicator of the complexity of a task and tells us how far we are from running our approach in Model Predictive Control (MPC) fashion.

First, we measure convergence rates. To obtain comparable results, we initialize all tasks with a stance controller and normalize the costs with the initial cost of each task. The results are shown in Figure 11. The curves suggest that there is a relation between the complexity of a task and its convergence rate. The trotting is a complex behavior with a long time horizon which might explain slow convergence. In contrast, the rearing task is relatively simple and converges quickly. When it comes to runtime, SLQ has a major advantage over other TO approaches: The complexity scales linearly with discretization steps and thus with time horizon. Figure 12 shows the runtime as a function of time steps, underlining this linear relation. The timings are measured on a quadcore laptop computer and averaged over all iterations for each task as indicated in Figure 11. To achieve these runtimes, we use a custom multi-threaded solver and optimized code for the system dynamics computation generated by RobCoGen [23]. Given the runtimes and the required iterations, we achieve convergence times of less than a minute for most tasks.

B. Robustness and Model Accuracy

There are various reasons why some of the motions and especially the gaits cannot yet be executed on hardware.

First, some tasks, such as the humanoid walking, exceed the physical limitations of the system. But since they illustrate the capabilities of the approach we still include them. Second, our TO is a deterministic approach without a notion of robustness. In some tasks robustness can be increased by cost function tuning. However, it remains an issue for all deterministic TO approaches since such methods cannot determine how sensitive a trajectory is to disturbance or model errors. E.g. the found trotting behavior would be more robust with an increased stepping frequency. Third, some tasks show slippage behavior leading to risky behaviors, potentially resulting in accidents. This slippage can have two sources: the friction cone and contact model softness. In the current form, there is no penalty for exceeding the friction cone. Therefore, sliding is not penalized or avoided, even when setting conservative limits. Additionally, the inherent trade-off in the contact model parameters between good gradients and physicality remains. Lastly, our control framework cannot modify step timings or locations to stabilize gaits efficiently. While heuristics like capture points methods exists, we think that running the approach in receding horizon fashion could address this issue in a more principled way.

VII. Conclusion and Outlook

We have presented a fully dynamic, whole-body TO framework able to create motions which involve multiple contact changes. The approach does not require any priors or initial guesses on contact points, sequences or timings. We demonstrate the capabilities on various tasks including squat jumps, rearing, balancing and rough manipulation. Furthermore, our TO is able to discover gaits such as galloping, trotting and two legged walking. Hardware results show that optimized trajectories can be transferred to physical systems. Despite the versatility of our approach, we obtain an optimized trajectory in less than one minute without warm starting. However, there are also some shortcomings. The solution space is huge, i.e. we can apply our TO to a broad variety of tasks. While this generality is a strength, it also requires to "choose" a solution by modifying cost function weights or adding additional terms. While this is a more or less intuitive approach, one would wish to further reduce the number of open parameters. Another drawback is that we can currently not transfer some motions onto hardware. We believe that running the approach as a model predictive controller will mitigate some of these issues. The presented timings suggest that - when warm starting and using shorter time horizons - SLQ is fast enough to be run with a receding horizon, even for complex systems such as HyQ. In previous work [22], we have already shown such an approach for stabilization, rapid replanning and disturbance rejection.

References

[1] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, 1998.

[2] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[3] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, p. 43, 2012.

[4] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *International Conference on Intelligent Robots and Systems*, 2012, pp. 1168–1175.

[5] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *International Conference on Robotics and Automation*, 2014, pp. 1168–1175.

[6] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *International Conference on Intelligent Robots and Systems*, 2015, pp. 3346–3351.

[7] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Hierarchical planning of dynamic movements without scheduled contact sequences," in *International Conference on Robotics and Automation*, 2016, pp. 4636–4641.

[8] I. Mordatch, K. Lowrey, and E. Todorov, "Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids," in *International Conference on Intelligent Robots and Systems*, 2015, pp. 5307–5314.

[9] C. Gehring, S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. A. Hoepflinger, and R. Y. Siegwart, "Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot," *Robotics & Automation Magazine*, vol. 23, no. 1, pp. 34–43, 2016.

[10] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *International Conference on Humanoid Robots*, 2014, pp. 279–286.

[11] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo, and J. Buchli, "Online walking motion and contact optimization for quadruped robots," in *International Conference on Robotics and Automation (to be published)*, 2017.

[12] D. Zimmermann, S. Coros, Y. Ye, R. W. Sumner, and M. Gross, "Hierarchical planning and control for complex motor tasks," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2015, pp. 73–81.

[13] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *International Conference on Robotics and Automation*, 2016, pp. 3555–3561.

[14] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[15] A. Sideris and J. E. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," *Transactions on Automatic Control*, vol. 50, no. 12, pp. 2043–2047, 2005.

[16] 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," *Institution of Mechanical Engineers, Journal of Systems and Control Engineering*, vol. 225, pp. 831–849, 2011.

[17] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference*, 2005, pp. 300–306.

[18] M. Neunert, F. Farshidian, and J. Buchli, "Efficient whole-body trajectory optimization using contact constraint relaxation," in *International Conference on Humanoid Robots*, 2016, pp. 43–48.

[19] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *International Conference on Robotics and Automation (to be published)*, 2017.

[20] T. Boaventura, "Hydraulic compliance control of the quadruped robot hyq," *PhD Thesis, University of Genova, Italy*, 2013.

[21] M. Bloesch, M. Hutter, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots – consistent fusion of leg kinematics and IMU," *Robotics: Science and Systems*, vol. 17, pp. 17–24, 2013.

[22] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *International Conference on Robotics and Automation*, 2016, pp. 1398–1404.

[23] M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini, "RobCoGen: a code generator for efficient kinematics and dynamics of articulated robots, based on Domain Specific Languages," *Journal of Software Engineering for Robotics*, vol. 7, no. 1, pp. 36–54, 2016.