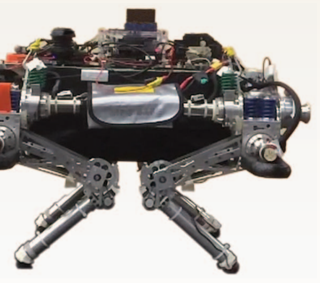
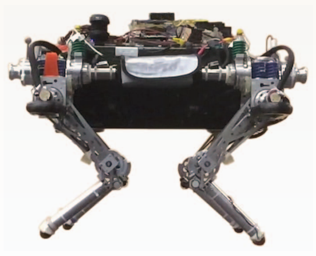
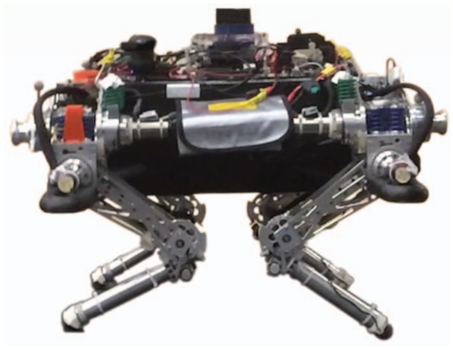
Practice Makes Perfect

An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot







By Christian Gehring, Stelian Coros, Marco Hutter, C. Dario Bellicoso, Huub Heijnen, Remo Diethelm, Michael Bloesch, Péter Fankhauser, Jemin Hwangbo, Markus A. Hoepflinger, and Roland Siegwart

This article approaches the problem of controlling quadrupedal running and jumping motions with a parameterized, model-based, state-feedback controller. Inspired by the motor learning principles observed in nature, our method automatically fine tunes the parameters of our controller by repeatedly executing slight variations of the same motion task. This learn-through-practice process is performed in simulation to best exploit computational resources and to prevent the robot from damaging itself. To ensure that the simulation results match the behavior of the hardware platform, we introduce and validate an accurate model of the compliant actuation system. The proposed method is experimentally verified on the torque-controllable quadruped robot StarlETH by executing squat jumps and dynamic gaits, such as a running trot, pronk, and a bounding gait.

Legged Locomotion

Legged locomotion enables humans and animals to traverse difficult environments with agility and grace. Given this remarkable ability, humans and animals have always served as a source of inspiration for the field of robotics. However, the locomotion skills of the state-of-the-art mobile robots are still limited compared with those seen in nature. Consequently,

Digital Object Identifier 10.1109/MRA.2015.2505910
Date of publication: 19 February 2016

1070-9932/16©2016IEEE

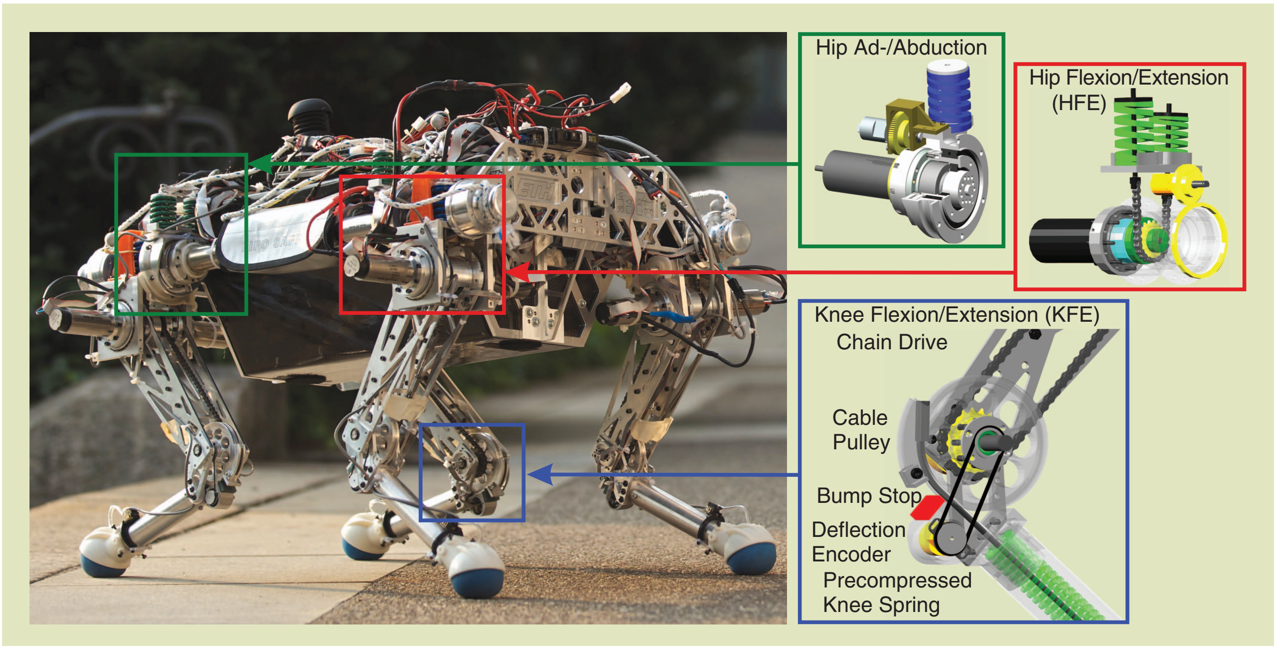
the problem of generating motor control behaviors for legged robotic systems remains a very active area of research.

To perform agile motions, especially those that might include airborne phases, a robot needs to be equipped with sufficiently powerful actuators. Fortunately, the development of high-performance drives for legged robots has seen significant progress in recent years. For instance, hydraulic systems, such as Boston Dynamic's quadrupeds [1] (BidDog, AlphaDog, and WildCat) and Italian Institute of Technology's HyQ [2], feature great performance and enable dynamic gaits, such as the flying trot, bound, and gallop. Likewise, research on electric actuators contributed to significant improvements in the motor skills of legged robots over the past few years. The Massachusetts Institute of Technology (MIT) Cheetah [3], for instance, uses electric motors with low gear reduction, and is skilled enough to bound over obstacles. Other systems employ biologically inspired actuators that make use of compliant elements, inspired by the fact that humans and animals leverage the elastic nature of their musculoskeletal structures [4]. By exploiting the compliance of the actuation system, the power and velocity output of electric motors can be amplified, as we showed for ScarlETH, a mechanical leg powered by series-elastic actuators (SEAs) [5] (Figure 1). A number of robots, including Marc Raibert's early machines [6], achieve dynamic maneuvers thanks to the compliance of the system.

As SEAs combine many features essential for legged robots, such as torque control, lightweight design, and robustness against impacts, we built a fully articulated quadruped robot, StarlETH [7], with highly compliant SEAs. To increase StarlETH's repertoire of motion skills, we are interested in developing flexible control strategies for various agile maneuvers, including running and jumping. One important main goal of our control strategy is to take the full advantage of the actuators' compliance.

Raibert's seminal work [6] on monopod hoppers with telescopic legs showed that a simple collection of control rules is applicable to a large set of dynamic motions and robots. Inspired by nature, contact force profiles were manually designed to enable the MIT Cheetah robot [3] to bound over obstacles. Alternatively, optimization algorithms can be used to generate various locomotion tasks by automatically finding appropriate values for parameterized control policies. Such direct policy search methods have been used, for instance, to generate galloping motions in simulation [8], to control muscle activations for simulated bipeds [9], to stabilize a planar bipedal robot [10], and to generate leaping motions for the wheeled quadruped robot PAW [11]. In previous work, we also applied direct policy search methods for motion synthesis. In particular, we employed PI^2 and ROCK* [12] to generate jumping and hopping maneuvers for our single-legged robot, ScarlETH.

Our work shows that policy search can be successfully applied to complex dynamical systems with a high-dimensional state and action space, enabling agile maneuvers. Our work is based on a generic set of motion primitives encoded through cost functions. Inspired by motor learning principles observed in nature, the cost functions are optimized through a trial-and-error process that requires the repeated execution of slight variations of a motion skill. Since executing these practice runs is time-consuming and could possibly damage the hardware, our goal is to utilize the policy search method in simulation only. However, to ensure that the optimized control policies also work on the hardware platform, we must bridge the gap between the simulation and the real world. Consequently, an accurate model of the robot is essential. In our previous work [13], we modeled the rigid-body dynamics of the quadruped, but neglected the SEAs. For highly agile motions, such as jumping, the actuator dynamics start to play



Hip Ad-/Abduction
Hip Flexion/Extension (HFE)
Knee Flexion/Extension (KFE)
Chain Drive
Cable Pulley
Bump Stop
Deflection Encoder
Precompressed Knee Spring

Figure 1. A quadruped robot StarlETH, equipped with SEAs, weighing 28 kg with a total leg length of 0.5 m. (Photo courtesy of François Pomerleau.)

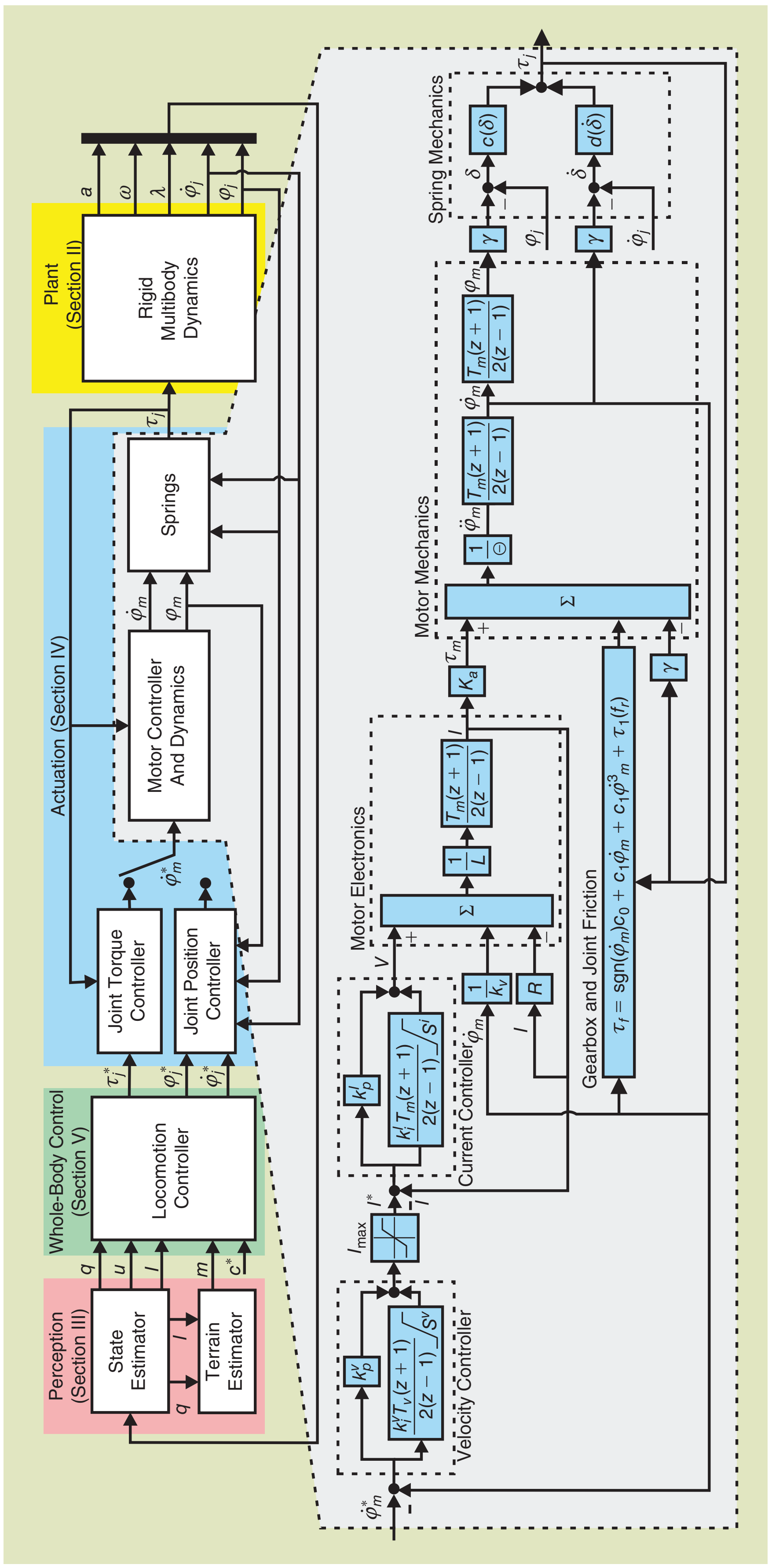


Figure 2. A block diagram of the controller and the plant.

a very important role and should not be ignored for two main reasons. First, the behavior predicted by simulation becomes much worse, and, therefore, the learned model may no longer perform well on the robot. Second, exploiting the dynamics of the actuators is crucial to ensuring optimal performance. The key to the success of our method relies on a model of the actuator unit that appropriately trades off accuracy versus computational complexity.

The control method that we describe in this article is designed to be applicable for various types of quadrupedal robots. Nevertheless, we make a few assumptions regarding actuation and perception capabilitie to generate stable locomotion skills. The block diagram shown in Figure 2 gives an overview of our controller and indicates the sections that describe each building block.

Mechanical Model of the Quadruped

The controller is primarily tailored to the mechanical model, as shown in Figure 3. The rigid multibody system is composed of 13 bodies, which are connected through three-actuated revolute joints per leg. This mechanical model is described with the generalized coordinates $\mathbf{q} \in \mathbb{R}^{15} \times SO(3)$, which include the pose of the torso with respect to the world frame and the joint angles. The generalized velocities $\mathbf{u} \in \mathbb{R}^{18}$ are the linear and angular velocities of the torso and the joint velocities.

The quadruped model is a nonsmooth scleronomic dynamical system and can be represented by a set of equations of the form

$$\mathbf{M}(\mathbf{q})\,\dot{\mathbf{u}} - \mathbf{h}(\mathbf{q},\mathbf{u}) - \mathbf{S}^\top \boldsymbol{\tau}_j - \mathbf{J}^\top \boldsymbol{\lambda} = \mathbf{0},\ \dot{\mathbf{q}} = \mathbf{F}(\mathbf{q})\,\mathbf{u}, \quad \boldsymbol{\gamma}_i = \mathbf{J}_i \mathbf{u} \ \ \forall i \in \mathcal{I}(\mathbf{q}). \qquad (1)$$

The mass matrix in (1) is denoted by **M**. The vector **h** covers the nonlinear terms, including Coriolis terms, centripetal terms, and impressed generalized forces, such as gravitational forces. The joint torques $\boldsymbol{\tau}_j$ are mapped to the generalized forces with the selection matrix $\mathbf{S}$, whereas the generalized velocities are mapped to the time derivative of the generalized coordinates with matrix $\mathbf{F}(\mathbf{q})$. The interaction between the rigid bodies is described by a set of closed contacts with the contact forces $\boldsymbol{\lambda} = [\ldots, \boldsymbol{\lambda}_i^\top, \ldots]^\top$ and the generalized force directions $\mathbf{J}^\top(\mathbf{q}) = [\ldots, \mathbf{J}_i^\top(\mathbf{q}), \ldots]$, where $\mathbf{J}_i = \partial \mathbf{r}_{WP_i}/\partial \mathbf{q}$ is the Jacobian of the corresponding contact point $\mathbf{r}_{WP_i}$. A contact i between two interacting bodies is said to be active if it is closed on displacement level, i.e., it is in $\mathcal{I}(\mathbf{q}) := \{i \mid g_{N_i}(\mathbf{q}) = 0\}$, where $g_{N_i}(\mathbf{q})$ describes the displacement of the two bodies in normal direction of a contact i. Set-valued force laws define the relation between the relative velocity $\boldsymbol{\gamma}_i(\mathbf{q})$ of a closed contact i and the corresponding generalized force $\boldsymbol{\lambda}_i$.

To prohibit a foot from penetrating and pulling on the ground, the (closed) contact is modeled as a unilateral contact, which can be represented by a normal cone inclusion on velocity level [14].

The frictional contact between the feet and the terrain is modeled as spatial Coulomb friction with a set-valued force law [14]. The applied force law represents slipping and sticking effects, which are characterized by the single friction parameter μ.

To include impacts in the equations of motion in (1), the set-valued force laws have to be supplemented with an impact law. For each normal cone inclusion, we use a Newton-type impact law [14] and assume a single contact point per foot with an inelastic impact. We note that this hard contact model is a physically accurate approximation of the real foot of StarlETH, which is spherical and only slightly compliant.

To solve the equations of motion with set-valued contact laws in (1) together with both the impact and impact-free motions, we employ a time-stepping scheme of Moreau [14]. A large benefit of this simulation method is that the time step between two successive simulation updates can be chosen to be relatively large without the simulation becoming unstable. For the modeled quadruped, the simulation time step can be set equal to the control update step of 2.5 ms. In addition, the simulation results are physically accurate [14], which is particularly essential when they are used in optimizations.

Perception

Advanced model-based locomotion controllers rely on fast and accurate feedback of the robot's state, namely, the generalized coordinates $\mathbf{q}$ and velocities $\mathbf{u}$, as well as on information about its environment for motion planning (respectively, adaptation).

State Estimation

To acquire fast and precise robot state estimates, we rely on sensor fusion of data from an inertial measurement unit (IMU) at the main body as well as kinematic measurements of the joints. This is necessary, since temporal integration of the accelerations and rotational rates coming from the IMU results in state drift, and commonly available IMU complementary filters provide access to roll and pitch angle but do not yield velocity estimates. Moreover, the underlying assumption of constant acceleration done in a complementary filter can corrupt the estimation process, especially for dynamic maneuvers with legged robots.

We implement an extended Kalman filter that makes use of the fact that the point of contact with the ground is typically stationary with respect to some inertial coordinate frame. To this end, we directly use the velocity error at the contact points as innovation term within the Kalman filter update step. The IMU measurements are used for state prediction and a simple Mahalonobis-based outlier detection scheme for slipping point contacts can be implemented. Together, this leads to a very accurate and robust estimation of the main body velocity and orientation [15].

Terrain Estimation

A model of the terrain can be inferred from the relative foot position measurements together with the orientation of the body with respect to gravity from the state estimation. We model the local terrain as a ground plane, as shown in Figure 3, and estimate both the position and the orientation of the plane. This information is important for walking on the sloped terrain, particularly with steep inclinations, to be able to adapt the body to the terrain.

The plane is fitted through the stance legs by solving a least-squares problem [16]. However, considering only the current support legs is not an option for dynamic gaits as opposed to static gaits, which always have at least three legs in contact with the ground. We, therefore, propose employing a history of footholds to estimate the slope of the terrain [16]. To detect changing slopes rapidly, we only consider the last foothold of each leg.

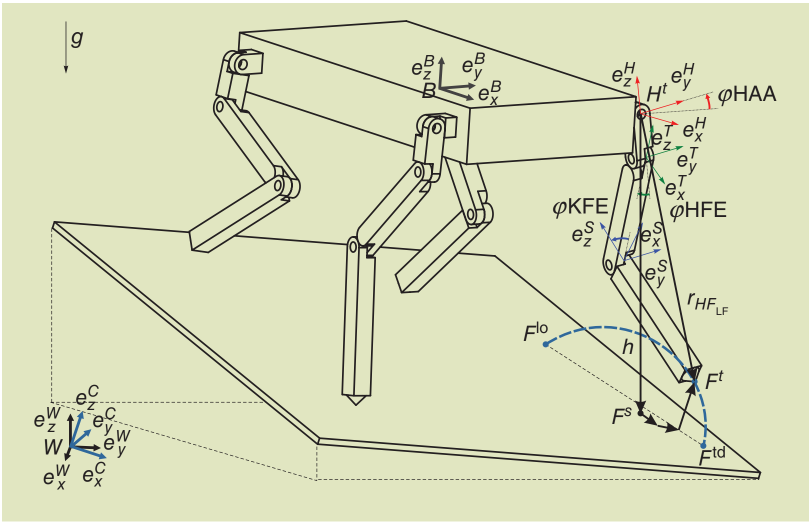


Figure 3. A rigid multibody model of the quadruped robot.

Actuation

StarlETH is equipped with SEAs in all joints, as shown in Figure 1, which enables both accurate joint position and torque tracking using the sophisticated feedback control on the joint level.

For applications that do not push the actuators to the limits [17], the actuator dynamics can be ignored, and the actuators can be considered as perfect torque and position sources. However, this idealization does not hold for highly dynamic maneuvers. In fact, various saturation and frictional effects can lead to a significant divergence of the commanded joint torque or position signals. To design a controller that takes these effects into account and even exploits the dynamics of the actuators (the compliance of the SEAs), a model of the actuation system is required.

A discrete-time model of the drive train of StarlETH is shown in Figure 2, which includes the following components.

- *Velocity controller*: The desired motor velocity $\dot{\varphi}_m^*$ is regulated by a proportional-integral controller with gains k_p^v and k_i^v together with an antiwindup, which saturates the state of the integrator at s^v. The loop is updated with time step $T_v = 1/1$ kHz and outputs the desired motor current I^*, which is limited to $I_{\max} = 9.4$ A on the motor drives.
- *Current controller*: A faster proportional-integral current controller with time step $T_m = 1/10$ kHz, gains k_p^I and k_i^I, and antiwindup saturation s^I determines the desired motor voltage V.
- *Motor electronics*: Motor resistance R and inductance L together with the back electromotive force $(1/\kappa_v \dot{\varphi}_m)$ are part of the motor κ_v electronics, which define the mapping of the applied voltage V to the motor current I. The dynamical effects of the power electronics and power supply are neglected. This loop is evaluated at the same rate as the current controller to minimize computational load.
- *Motor mechanics*: The motor current I multiplied with the torque constant κ_a yields the motor torque τ_m, which acts on the motor shaft together with the friction torque τ_f and the load at the joint $\gamma\tau_j$. The equations of motion of the motor shaft defining the motor velocity $\dot{\varphi}_m$ and position φ_m can be solved with the lumped motor, gearbox, and bearing inertia Θ.
- *Gearbox and joint friction*: The friction of the harmonic drive gearbox can be approximated with a constant (c_0), linear (c_1), and cubic (c_2) term of the motor velocity [18]. The frictional torque τ_l due to radial load f_r in the bearings is also affecting the dynamics of the actuator.
- *Spring mechanics*: The joint torque τ_j is a function of the spring deflection δ, the spring stiffness $c(\delta)$, the time derivative of the deflection $\dot{\delta}$, and the damping $d(\delta)$. For the detailed models of the spring characteristics, see [5].

Most parameters of the actuator model and the joint controllers are known from datasheets. The remaining parameters are identified based on different experiments [5].

The desired motor velocity $\dot{\varphi}_m^*$ is finally generated by the joint controllers, which regulate either the desired joint torque τ_j^* or the desired joint position φ_j^* and velocity $\dot{\varphi}_j^*$, as described in [5].

Locomotion Control

The (deterministic) locomotion control problem for periodic gaits, which minimizes a cost function c, can be stated as:

$$\begin{aligned} \mathbf{q}^*, \mathbf{u}^*, \boldsymbol{\tau}_j^*, \boldsymbol{\lambda}^*, \mathcal{I}^*(\mathbf{q}^*) &= \operatorname{argmin} c(\mathbf{q}, \mathbf{u}) \\ \text{s. t.} \quad \phi(t) = \phi(t+T) \quad \forall t, &\left.\right\} \text{periodicity} \\ \left.\begin{aligned} \mathbf{M}(\mathbf{q})\,\dot{\mathbf{u}} - \mathbf{h}(\mathbf{q},\mathbf{u}) - \mathbf{S}^\top \boldsymbol{\tau}_j - \mathbf{J}^\top \boldsymbol{\lambda} = \mathbf{0}, \\ \dot{\mathbf{q}} = \mathbf{F}(\mathbf{q})\mathbf{u}, \ \boldsymbol{\gamma}_i = \mathbf{J}_i \mathbf{u} \ \ \forall i \in \mathcal{I}(\mathbf{q}), \end{aligned}\right\} &\text{dynamics} \\ \mathbf{q}_j \in \mathcal{Q}, \quad \mathbf{u}_j \in \mathcal{U}, \quad \boldsymbol{\tau}_j \in \mathcal{T}, &\left.\right\} \text{limits}, \end{aligned} \tag{2}$$

where $\phi(t)$ is the periodic solution with a period time of T. The solution fulfills all dynamic constraints and does not violate any position limits $\mathcal{Q}$, velocity limits $\mathcal{U}$, or actuator limits $\mathcal{T}$. The periodic constraint can be replaced by the constraint $\phi(T) = \phi^*$ for nonperiodic motions like jumps.

In general, this is a hard control problem to solve because the motion planning/generation problem, which specifies how the legs and the main body should move $(\mathbf{q}^*, \mathbf{u}^*)$, and the motion execution/control problem, which determines the desired motion through the desired torques and forces $(\boldsymbol{\tau}_j^*, \boldsymbol{\lambda}^*)$, is closely coupled by the set of desired contacts $\mathcal{I}^*(\mathbf{q}^*)$. Moreover, robust locomotion is only achieved if the controller is capable to handle large disturbances due to unanticipated terrain irregularities or external pushes. As a consequence, motor actions cannot be planned fully offline and tracked in an open-loop fashion, but have to be generated online by a combination of a feedforward and state-feedback controller instead.

We employ a model-based controller with a compact, yet flexible parameter space and fine-tune these parameters by repeating the same motion task with slight parameter variations until a desired motion is found and all constraints are met. The proposed controller generates a desired motion based on predefined motion primitives, which are superimposed by control actions needed for balancing. To find these motion primitives , along with other parameters, we reformulate the problem stated in (2) to

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathcal{P}}{\operatorname{argmin}} \sum w_k c_k(\mathbf{q}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\varphi}_m, \dot{\boldsymbol{\varphi}}_m), \tag{3}$$

where the optimal parameter set $\boldsymbol{\theta}^*$ in the admissible set $\mathcal{P}$ minimizes the weighed sum of the cost terms c_k, which depend on the state trajectory of the robot $\mathbf{q}$, $\mathbf{u}$ and the actuation signals $\boldsymbol{\varphi}_m, \dot{\boldsymbol{\varphi}}_m$.

Motion Parameterization

Contact Scheduling

The desired motions of the legs need to be parameterized in space and time. We split the motions into individual motion primitives according to the contact schedule, which defines the set of contacts $\mathcal{I}^*(\mathbf{q}^*)$ and the role of the legs between the contacts. A stance leg supports the main body, whereas a swing leg moves its foot to a new location. A gait pattern can be employed to predefine the contact schedule. For periodic motions, the gait pattern is defined as nine parameters, which include liftoff and landing time for each leg as well as the stride duration T that defines the period of the cycle. We use a reduced parameterization that can represent both symmetrical and asymmetrical gaits that allows us to define the parameter space for a specific type of gait, which helps to optimize the time parameters for a predefined gait [13].

Terrain Adaptation

The motions of the legs need to be planned and replanned according to the encountered terrain. We introduce a control frame C, as shown in Figure 3, to properly define the motions on the arbitrary terrain. The z-axis of the control frame ($\mathbf{e}_z^C$) is aligned with the estimated surface normal, and the x-axis ($\mathbf{e}_x^C$) is parallel to the projected x-axis of the body fixed base frame B on the ground. The origin of the coordinate system C is fixed to the origin of the world frame W such that only the orientation of the frame is changing over time.

By describing the desired motion (e.g., body orientation) in this control frame, the motion is properly defined and automatically aligned with the terrain. The same holds true for high-level velocity commands $\mathbf{c}^* = [v_x, v_y, \dot{\psi}]^\top$, which are typically given in heading direction ($v_x \mathbf{e}_x^C$) or lateral direction ($v_y \mathbf{e}_y^C$) of the robot, as well as desired turning rate $\dot{\psi}\mathbf{e}_z^C$ around the vertical axis.

Motion of Stance Legs

Since the configurations of all support legs are defined as the fixed foothold locations and the pose of the torso, we define the motions of all support legs through a desired motion of the torso. For balancing, the center of mass is simply kept above the support region. By averaging the position of the legs with weights depending on their role, a smooth motion of the target of the center of mass is planned [19]. The remaining degrees of freedom (DoF) like body height above ground and orientation are prescribed by motion primitives. We employ polyharmonic and quintic polynomial splines with periodic constraints to define the feedforward motion of the main body. To reduce kinematic singularities, the default orientation of the torso is adapted to the local inclination of the terrain and superimposed by gait-specific motions, such as pitching for a bounding gait.

Motion of Swing Legs

The desired footpoint F^t of a swing leg is defined as the sum of several position vectors, as shown in Figure 3. The basic idea is to plan a desired foothold F^{td} at the end of the swing phase and then to interpolate between liftoff position F^{lo} and this position at touch-down as a function of the swing phase. For the ground clearance, a trajectory in the z-direction of the control frame C is defined using a spline with zero velocity constraints at the beginning and ending of the swing phase to avoid step inputs to the actuators and minimize impact losses.

The desired foothold location F^{td} is determined at every control step based on a desired feedforward motion and a superimposed balancing controller. First, a location on the slope F^s is selected with respect to the hip joint H^t. For locomotion on the sloped terrain, there are two different strategies: 1) the lever mechanism and 2) the telescopic strut. The lever mechanism projects the hip position along the slope normal, whereas the telescopic strut strategy projects the hip position along the gravity. We apply the latter strategy as discussed in [16]. From the selected location on the ground, the desired foothold is defined as the desired traveling distance given by the speed command and the timing given by the gait pattern. In presence of a disturbance, the location of the foothold is corrected by the balancing controller. We employ an adapted version of Raibert's flight controller [6] for each leg, which predicts the next foothold according to

$$k_p^{\text{FB}}(\mathbf{v}_{\text{ref}}^* - \mathbf{v}_{\text{ref}})\sqrt{\frac{h}{g}}, \tag{4}$$

where $\mathbf{v}_{\text{ref}}^*$ is the desired and $\mathbf{v}_{\text{ref}}$ is the measured reference velocity between associated hip and middle of the torso, h is the height of the hip above the ground, and g is the gravitational acceleration. The component of the balancing control is weighted by k_p^{FB} and only active if there is a velocity error.

Motion Execution

Leg Coordination

A leg coordinator decides based on the planning given by the gait pattern and contact sensing if a leg is considered to be a support leg and, thus, is force controlled or if it is a swing leg and position control can be safely used. We further employ an event detector, which identifies for each leg events such as early and late touchdown, early and late liftoff, slipping contact, lost contact during stance, and hitting an obstacle during swing to trigger different reflex mechanisms.

No special treatments are required for late liftoff and early touchdown. In case of slipping and lost contact during the stance phase, the leg is lowered with respect to the ground plane to regain contact as quickly as possible.

Execution of Swing Leg Motion

The desired swing foot positions are enforced by mapping the Cartesian positions to the desired joint positions $\boldsymbol{\varphi}_j^*$ and velocities $\dot{\boldsymbol{\varphi}}_j^*$ using inverse kinematics, which are subsequently tracked by the joint position controller.

Execution of Torso Motion

We apply a virtual model controller in combination with a quasi-static force distribution to track the desired motion of the main body while optimally distributing the ground reaction forces. The virtual model controller outputs a desired net force $\mathbf{f}^*$ and torque $\mathbf{t}^*$, which should act on the torso to execute the desired motion defined as the desired position $\mathbf{r}_{\text{WB}}^*$, velocity $\mathbf{v}_B^*$, and acceleration $\mathbf{a}_B^*$ as well as desired rotation quaternion $\mathbf{p}_{\text{WB}}^*$ and angular velocity $\boldsymbol{\omega}_{\text{WB}}^*$. The controller has the form of

$$
\begin{aligned}
\mathbf{f}^* &= \mathbf{K}_p^f(\mathbf{r}_{\mathrm{WB}}^* - \mathbf{r}_{\mathrm{WB}}) + \mathbf{K}_i^f \int (\mathbf{r}_{\mathrm{WB}}^* - \mathbf{r}_{\mathrm{WB}}) \\
&\quad + \mathbf{K}_d^f(\mathbf{v}_B^* - \mathbf{v}_B) + \mathbf{a}_B^* m - \sum_{k \in \mathcal{B}} \mathbf{f}_k^g \\
\mathbf{t}^* &= \mathbf{K}_p^t(\mathbf{p}_{\mathrm{WB}}^* \boxminus \mathbf{p}_{\mathrm{WB}}) + \mathbf{K}_i^t \int (\mathbf{p}_{\mathrm{WB}}^* \boxminus \mathbf{p}_{\mathrm{WB}}) \\
&\quad + \mathbf{K}_d^t(\boldsymbol{\omega}_{\mathrm{WB}}^* - \boldsymbol{\omega}_{\mathrm{WB}}) - \sum_{k \in \mathcal{B}} \hat{\mathbf{r}}_{\mathrm{WS}_k} \mathbf{f}_k^g,
\end{aligned} \tag{5}
$$

where $\mathbf{f}_k^g$ is the gravitational force acting on body k, m is the mass of the main body, $\mathbf{K}$ is the diagonal gain matrices, $\hat{\mathbf{r}}_{\mathrm{WS}_k}$ is the skew-symmetric matrix defining the cross product of the position vector $\mathbf{r}_{\mathrm{WS}_k}$ from the the origin of the world frame to the center of mass of body k, and $\boxminus: SO(3) \times SO(3) \to \mathbb{R}^3, q_1, q_2 \mapsto \log(q_1 \otimes q_2^{-1})$. We note that the integrators are important to compensate for modeling errors, especially for jumping motions where it is important to launch the maneuver from a predefined state. All integrator states are limited to account for windup.

The desired net force and net torque need to be generated by the contact forces $\boldsymbol{\lambda}^*$. To this end, the *force distribution* computes the desired contact forces for each support leg i while accounting for the contact constraints $(\mu, \lambda_{N,\min})$ and torque limits $(\tau_{\min}, \tau_{\max})$ for each joint j. By approximating the friction cones as pyramids, the problem can be formulated as the quadratic program:

$$
\begin{aligned}
\boldsymbol{\lambda}^* = \operatorname{argmin} (\mathbf{A}\boldsymbol{\lambda} - \mathbf{b})^\top \mathbf{S} (\mathbf{A}\boldsymbol{\lambda} - \mathbf{b}) + \boldsymbol{\lambda}^\top \mathbf{W} \boldsymbol{\lambda} \\
\left.\begin{aligned}
\underbrace{\begin{bmatrix} \cdots & \mathbf{I} & \cdots \\ \cdots & \hat{\mathbf{r}}_{\mathrm{BF}_i} & \cdots \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \vdots \\ \boldsymbol{\lambda}_i \\ \vdots \end{bmatrix}}_{\boldsymbol{\lambda}} = \underbrace{\begin{bmatrix} \mathbf{f}^* \\ \mathbf{t}^* \end{bmatrix}}_{\mathbf{b}} \\
\text{s. t. } \lambda_{N,i} \geq \lambda_{N,\min}, \quad \| \boldsymbol{\lambda}_{T,i} \| \leq \mu \lambda_{N,i}, \\
\tau_{\min} \leq \tau_{j,j}(\lambda) \leq \tau_{\max}
\end{aligned}\right\} \forall_i \in \mathcal{I}(\mathbf{q}),
\end{aligned} \tag{6}
$$

where $\mathbf{r}_{\mathrm{BF}_i}$ is the position vector from base to foot. The diagonal matrix $\mathbf{S}$ in (6) weights distinct DoF with different weights and is used to balance the different scales implied by the different units. The diagonal matrix $\mathbf{W}$ in (6) is used as regularizer, which tries to minimize the contact forces.

The desired joint torques are finally obtained by simple Jacobi-transposed mapping

$$
\boldsymbol{\tau}_j^* = -\sum_{i \in \mathcal{I}(\mathbf{q})} \mathbf{J}_i^\top \boldsymbol{\lambda}_i^* - \sum_{k \in \mathcal{B}} \mathbf{J}_k^\top \mathbf{f}_k^g, \tag{7}
$$

with $\mathbf{J}_k = \partial \mathbf{r}_{\mathrm{BS}_k} / \partial \mathbf{q}$ for body k. We note that by considering the contact constraints, this approach significantly increases safety against slippage, particularly when operating in the sloped terrain.

Motion Learning

Optimization Method

To optimize the behavior of the robot, we use the covariance matrix adaptation evolution strategy (CMA-ES) of Hansen [20], which is the state-of-the-art sampling-based black box optimization algorithm and well suited for our high-dimensional, nonlinear, and nonsmooth problem. The evolutionary algorithm is a local minimizer, which requires a good initial guess.

The CMA-ES samples new parameters from a multivariate Gaussian distribution and does not take the admissible parameter space $\mathcal{P}$ into account. We, therefore, apply a combination of a rejection and a projection sampling strategy. If a parameter lies outside of the permissible set, a new sample is drawn until a valid set is found. After a 1,000 trials, the invalid sample is projected to the boundary of $\mathcal{P}$, and a cost proportional to the distance to the boundary is added to the total cost.

We apply a staged optimization process, which starts with a manually tuned initial guess. We first optimize the motions with perfect state knowledge and relaxed joint and actuator limits. Then, we gradually increase the complexity of the model, until we are sure that we can apply the found control parameter set on the real robot.

Squat Jump

A simple locomotion task is to perform a squat jump, which is when the robot starts in a crouched position with all legs on the ground. During the launch phase, the robot accelerates its main body in the vertical direction to jump as high as possible. The legs should be retracted to achieve a large ground clearance during the flight phase. Before landing, the legs should be extended again such that the robot lands safely without slipping and rebounding.

We parameterize the desired height trajectory of the torso with a quintic spline with eight knots evenly distributed over a fixed time window of 0.4 s and the height of the feet with respect to the main body during the flight phase with a spline with three knots for the left fore foot, which is then mirrored for the other feet. The feet are positioned under the hips during the whole jump. The initial height of the torso and the damping gain in the direction of gravity of the virtual model controller are added to the optimization variables.

The optimization goal is to maximize the apex height of the jump by minimizing the height of the torso in the world frame: $-|\max_{t \in [0,T]} ({}_W\mathbf{r}_{\mathrm{WB}}(t)^\top \mathbf{e}_z^W)|$. A large ground clearance between the feet and the ground is achieved by adding a cost term that minimizes the leg length during the flight phase. The velocities at the feet, which are in contact with the ground, are also minimized to avoid slippage. Furthermore, the time spent during a rebound is encoded in a cost function. The force distribution filters the desired net force $\mathbf{f}^*$ and torque $\mathbf{t}^*$ based on the joint torque limits and contact constraints. This filter has the effect that different input signals result in the same cost values. To make CMA-ES aware of this behavior, we add the tracking error to the cost function. In addition, contact forces $\boldsymbol{\lambda}$ close to the boundary of the friction cone are penalized to account for tracking errors due to the actuator dynamics.

Dynamic Gaits

To find the optimal parameter set for a specific dynamic (periodic) gait, it is essential to optimize the motions together with the timing parameters of the gait pattern. To find a running trot with a long flight phase, we optimize for the cycle duration and the duty factor, which defines for a leg how much time it is supposed to be on the ground, together with the parameters defining the height trajectory of the torso. In addition to gait-specific cost terms like maximal flight duration for the running trot, we use similar cost terms as for the squat jump to find a feasible motion. To promote stable and feasible motions, the simulated quadruped has to walk several gait cycles and withstand external disturbances without falling on the ground. More details about tuning control parameters for dynamic gaits can be found in [13].

Results

The presented locomotion controller was implemented on StarlETH and enabled the robot to perform periodic gaits, such as static walk, dynamic trot, or bound, and highly dynamic maneuvers, such as a vertical jump. The robot was

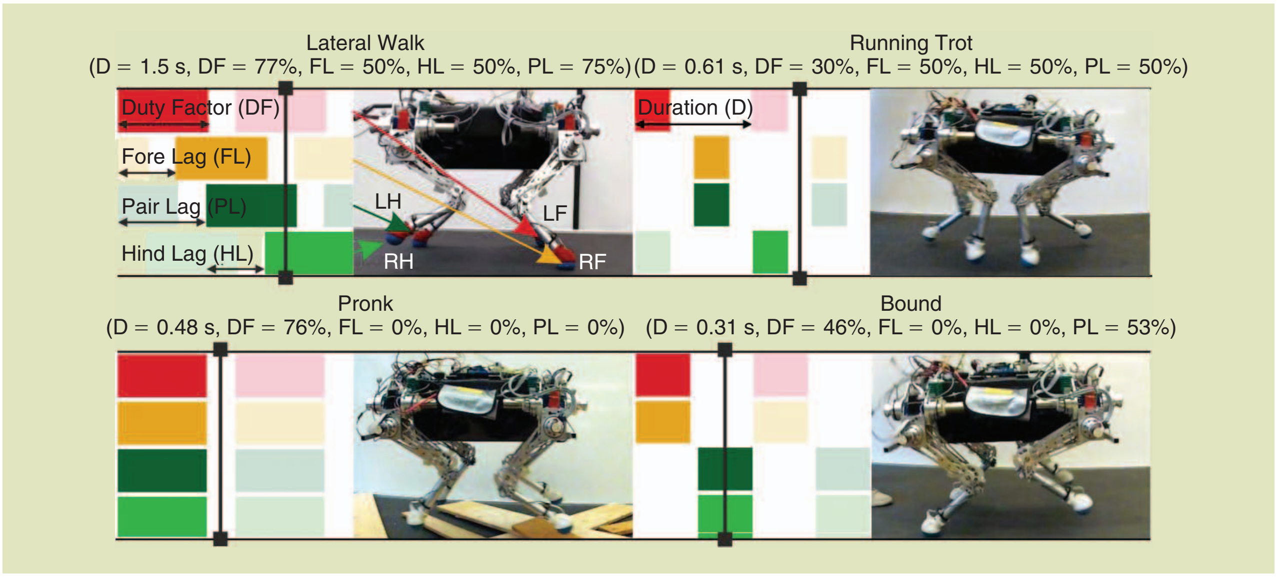

Lateral Walk
(D = 1.5 s, DF = 77%, FL = 50%, HL = 50%, PL = 75%)
Running Trot
(D = 0.61 s, DF = 30%, FL = 50%, HL = 50%, PL = 50%)
Duty Factor (DF)
Fore Lag (FL)
Pair Lag (PL)
Hind Lag (HL)
LH
LF
RH
RF
Duration (D)
Pronk
(D = 0.48 s, DF = 76%, FL = 0%, HL = 0%, PL = 0%)
Bound
(D = 0.31 s, DF = 46%, FL = 0%, HL = 0%, PL = 53%)

Figure 4. The gait patterns for various gaits are shown together with snapshots of StarlETH. The colored bars indicate if a leg is supposed to be in stance (adapted from [13]).

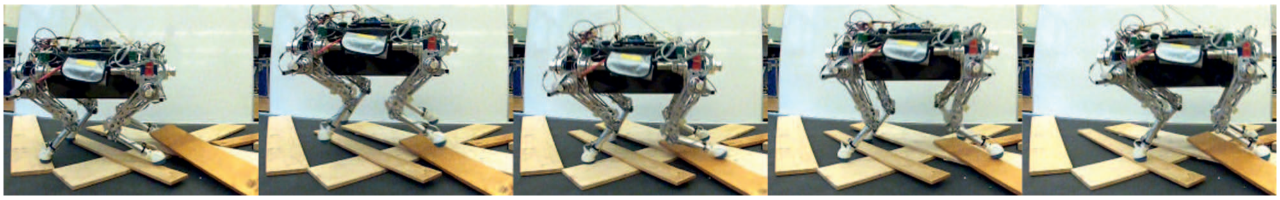


Figure 5. The StarlETH performs a pronking gait while dealing with unperceived obstacles. (adapted from[13]).

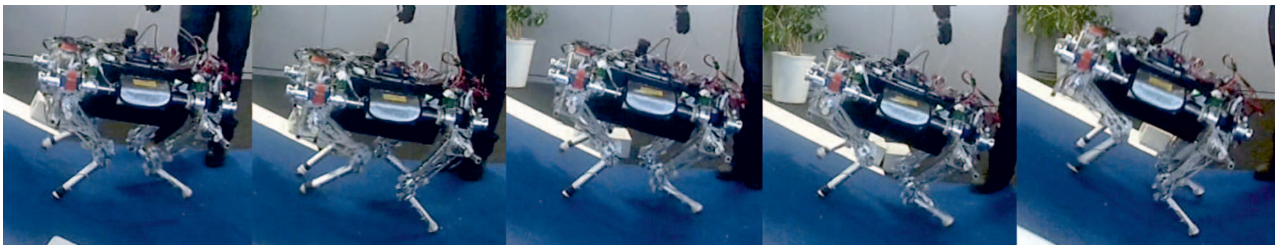


Figure 6. The StarlETH transits from flat to the sloped terrain with 21° (38%) while trotting (adapted from [16]).

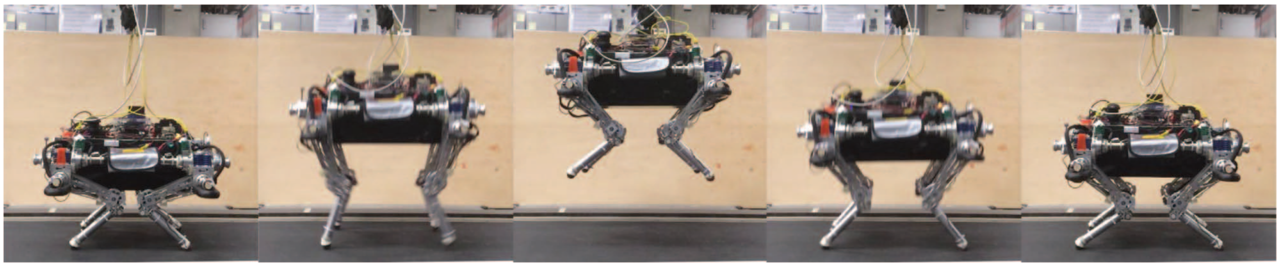


Figure 7. The StarlETH performs a vertical jump: start position, takeoff, apex height, landing, and end position.

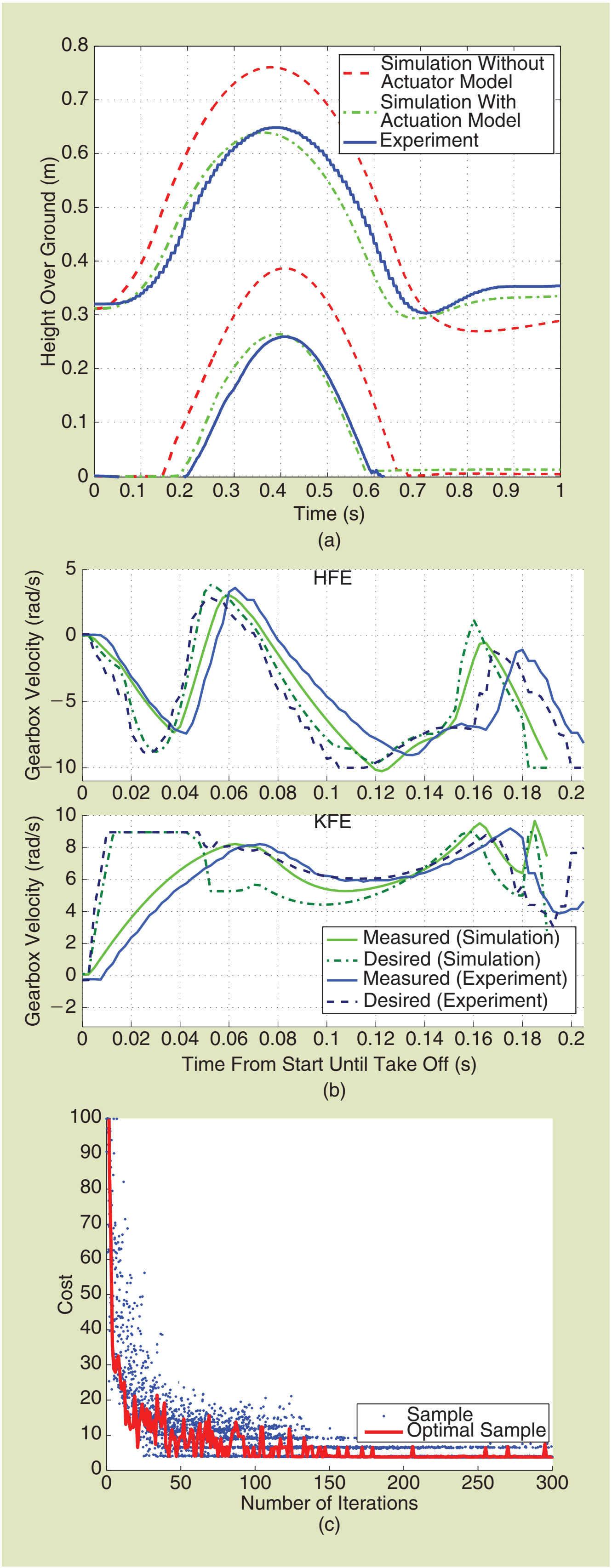


Figure 8. The results of a vertical jump. (a) The height trajectories of torso and foot. (b) The gearbox velocities during the launch phase. (c) The convergence of the cost.

completely autonomous (onboard power and computation) during all these maneuvers.

Periodic Gaits

The sampling-based method successfully tuned 18 control parameters for trotting, bounding, and pronking gaits in simulation. Due to reasonably good accordance of the model, the parameters could be directly used on the real platform. The found gait patterns and some snapshots of the real robot performing these gaits are shown in Figure 4 (video: http://youtu.be/Tj1wreifYhU). A time series of snapshots is shown in Figure 5, which captures two cycles of a pronking gait.

The proposed controller is robust enough to deal with unperceived obstacles up to 10% of the leg length (video: http://youtu.be/Wuc7mL0hkGo), as well as with external pushes up to 100 N, as shown in [7].

Furthermore, the terrain estimation together with the adaption of the control signals to the modeled ground plane allowed StarlETH to trot on slopes (video: http://youtu.be/NPuHwxpVUpg) with an angle of up to 21° (38%), as shown in Figure 6. A 360° turn on the slope verified that the slope estimation works as intended [16].

We want to particularly highlight here that we were not able to find control parameters for highly dynamic gaits by hand despite significant effort and extensive experience (and, hence, intuition) with the machine. However, the optimization approach helped to find a feasible set for more dynamic gaits, such as a running trot, pronk, and bound.

High-Dynamic Maneuvers

A sequence of the optimized vertical jump (video: http://youtu.be/aEsxLN9CnyE) is shown in Figure 7 (video: http:// youtu.be/aEsxLN9CnyE). The robot's main body reached the apex height of 0.76 m, which corresponds to 150% of the leg length.

The height trajectories of the torso and the left forefoot are shown in Figure 8(a). The torso height was measured by an external motion capture system (Optitrack), whereas the foot height measurement is based on the state estimation.

To generate this high-dynamic jump whereby the motors encounter several saturation effects, it was particularly important to have an accurate actuator model. As a result, the trajectory, which was optimized in simulation with the actuator model, comes close to the trajectory of the real robot. If the same optimized desired trajectories are applied in simulation without the actuator model, the simulated robot jumps significantly higher than StarlETH.

Good agreement between the measured and predicted gearbox velocities together with the commands from start until the takeoff of the jump is shown in Figure 8(b).

The convergence of the optimization is shown in Figure 8(c). After 250 iterations with a population of 15 samples, the best ever seen cost was found.

Conclusion

Inspired by the principle of learning through practice, we applied a sampling-based search to optimize locomotion controllers that significantly expand the repertoire of motion skills for StarlETH, our quadruped robot. We addressed the high-dimensional, nonsmooth nature of the locomotion control problem by finding a good tradeoff between feedforward motion primitives, which are optimized based on the design of the robot, and robust state-feedback control, which compensates for modeling errors and sensor noise and rejects unanticipated disturbances. We showed that by appropriately modeling the contact and actuator dynamics of our compliant quadruped, the simulation-based optimization results were directly applicable to the real platform. Based on our results, we believe that our approach constitutes an important step toward a fully automatic solution for generating behaviors and motion skills for a wide variety of legged robots. To promote further work in this area, we have open-sourced our implementation of the state estimator and motion controller, both of which can be found on our website: http:// leggedrobotics.ethz.ch.

References

[1] M. Raibert, "BigDog, the rough-terrain quadruped robot," in Proc. *IFAC Volumes (IFAC-PapersOnline)*, 2008, vol. 17, pp. 6–9.

[2] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *Proc. IEEE Int. Conf. Robotics Automation*, 2013, pp. 2554–2561.

[3] H.-W. Park, M. Yee, M. Chuah, and S. Kim, "Dynamic quadruped bounding control with duty cycle modulation using vertical impulse scaling," in *Proc. IROS*, 2014, pp. 3–4.

[4] G. A. Cavagna, N. C. Heglund, and C. R. Taylor, "Mechanical work in terrestrial locomotion: Two basic mechanisms for minimizing energy expenditure," *Amer. J. Physiol. Regul., Integr. Comparative Physiol.*, vol. 233, pp. 243–261, Nov. 1977.

[5] M. Hutter, C. D. Remy, M. A. Hoepflinger, and R. Siegwart, "ScarlETH: Design and control of a planar running robot," in *Proc. IEEE Int. Conf. Intelligent Robots Systems*, Sept. 2011, pp. 562–567.

[6] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA: Massachusetts Institute of Technology, 1986.

[7] M. Hutter, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Towards combining speed, efficiency, versatility and robustness in an autonomous quadruped," *IEEE Trans. Robotics (under review)*, vol. 30, no. 6, pp. 1427–1440, 2014.

[8] D. P. Krasny and D. E. Orin, "Evolution of dynamic maneuvers in a 3D galloping quadruped robot," in *Proc. IEEE Int. Conf. Robotics Automation*, May 2006, pp. 1084–1089.

[9] T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *ACM Trans. Graph.*, vol. 32, no. 6, 2013, pp. 206:1–206:11.

[10] B. Griffin and J. Grizzle, "Walking gait optimization for accommodation of unknown terrain height variations," in *Proc. American Control Conf.*, 2015, pp. 4810–4817.

[11] A. Harmat, M. Trentini, and I. Sharf, "Jumping behaviour for a wheeled quadruped robot: Simulation and experiments," *J. Unmanned Veh. Syst.*, vol. 60, pp. 41–60, Oct. 2013.

[12] J. Hwangbo, C. Gehring, H. Sommer, R. Siegwart, and J. Buchli, "ROCK*—Efficient black-box optimization for policy learning," in *Proc. 14th IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, Nov. 2014, pp. 535–540.

[13] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Towards automatic discovery of agile gaits for quadrupedal robots," in *Proc. IEEE Int. Conf. Robotics Automation*, 2014, pp. 4243–4248.

[14] C. Gehring, G. Nuetzi, R. Diethelm, R. Siegwart, and R. I. Leine, "An evaluation of Moreaus time-stepping scheme for the simulation of a legged robot," in *Proc. ASME Int. Design Engineering Technical Conf.*, 2014.

[15] 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 *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, Nov. 2013, pp. 6058–6064.

[16] C. Gehring, C. D. Bellicoso, S. Coros, M. Bloesch, P. Fanhauser, M. Hutter, and R. Siegwart, "Dynamic trotting on slopes for quadrupedal robots," in *Proc. IEEE Int. Conf. Intelligent Robots Systems*, 2015, pp. 5129–5135.

[17] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *Int. J. Robot. Res.*, vol. 33, no. 8, pp. 1047–1062, May 2014.

[18] T. D. Tuttle and W. P. Seering, "A nonlinear model of a harmonic drive gear transmission," *IEEE Trans. Robotics Automation*, vol. 12, no. 3, pp. 368–374, 1996.

[19] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of dynamic gaits for a quadrupedal robot," in *Proc. IEEE Int. Conf. Robotics Automation*, May 2013, pp. 3287–3292.

[20] N. Hansen, "The CMA evolution strategy: A comparing review," *Stud. Fuzziness Soft Comput.*, vol. 192, no. 2006, pp. 75–102, 2006.

Christian Gehring, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: gehrinch@ethz.ch.

Stelian Coros, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States. E-mail: scoros@andrew.cmu.edu.

Marco Hutter, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: mahutter@ethz.ch.

C. Dario Bellicoso, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: bellicosodario@gmail.com.

Huub Heijnen, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: hheijnen@student.ethz.ch.

Remo Diethelm, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: rdiethelm@gmail.com.

Michael Bloesch, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: bloeschm@ethz.ch.

Péter Fankhauser, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: pfankhauser@ethz.ch.

Jemin Hwangbo, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: jhwangbo@ethz.ch.

Markus A. Hoepflinger, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: hoepflinger@mavt.ethz.ch.

Roland Siegwart, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: rsiegwart@ethz.ch.