

Control of Dynamic Gaits for a Quadrupedal Robot

Christian Gehring^{*†}, Stelian Coros[†], Marco Hutter^{*},
Michael Bloesch^{*}, Markus A. Hoepflinger^{*} and Roland Siegwart^{*}
^{*}Autonomous Systems Laboratory, ETH Zurich, Switzerland, gehrinch@ethz.ch
[†]Disney Research Zurich, Switzerland

Abstract—Quadrupedal animals move through their environments with unmatched agility and grace. An important part of this is the ability to choose between different gaits in order to travel optimally at a certain speed or to robustly deal with unanticipated perturbations. In this paper, we present a control framework for a quadrupedal robot that is capable of locomoting using several gaits. We demonstrate the flexibility of the algorithm by performing experiments on StarlETH, a recently-developed quadrupedal robot. We implement controllers for a static walk, a walking trot, and a running trot, and show that smooth transitions between them can be performed. Using this control strategy, StarlETH is able to trot unassisted in 3D space with speeds of up to 0.7m/s, it can dynamically navigate over unperceived 5-cm high obstacles and it can recover from significant external pushes.

I. INTRODUCTION

Legged robots are better suited for rough terrain locomotion than their wheeled or tracked counterparts. As a result, they have the potential of being used for a wider variety of tasks. The drawback of multi-legged systems, however, is that they are more complex, inherently unstable and therefore more difficult to control. In addition, appropriate control methods need to be robust to unplanned disturbances because the environments, in general, are only partially observable. Statically stable solutions for this problem rely on position control algorithms and have been studied extensively [1], [2]. However, they have not yet been shown to produce motions that are as agile as the motions observed in nature [3].

To date, considerable progress has been made towards bridging the gap between the skill sets of legged robotic systems and that of real animals. In contrast to Boston Dynamic's LittleDog [4], which is only capable of static walking, its larger counterpart, BigDog [5], looks more agile and life-like and is capable of a variety of locomotion behaviors: standing up, squatting down, walking, trotting and bounding. Stable and robust locomotion has been demonstrated on this platform, but the exact details of the employed control algorithms are not known. More information is available regarding the control scheme of the IIT's HyQ [6], another hydraulically actuated quadruped, which was recently shown trotting robustly by employing a simple virtual model control approach for each leg [7].

Recent progress has also been made in simulation, where it is possible to decouple the control laws from the limitations of specific hardware platforms. For instance, Coros et al. [8]

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

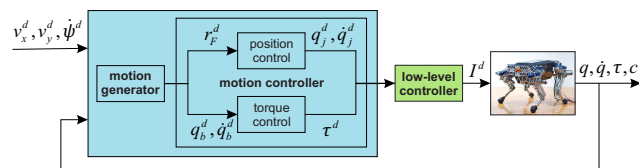


Fig. 1. The control framework (blue) computes the desired joint positions, velocities, and torques based on the desired walking speed and direction, whereas the low-level controller (green) generates the desired motor current from these outputs.

described a control framework that was successfully used to control a broad range of dynamic gaits for a dog-like simulated quadruped, and Krasny and Orin [9] developed a control algorithm for galloping quadrupeds.

At a high level, successful control strategies are based on the observations noted by Raibert [10], who showed the importance of two essential ingredients: control of the body through the stance hips and foot placement control for balance recovery.

For this work we use a set of conceptually similar ideas in order to control StarlETH (Springy Tetrapod with Articulated Robotic Legs), a recently developed, electrically driven quadruped robot [11]. StarlETH's weight of 23kg and leg length of 0.4m correspond to the dimensions of a medium sized dog, and it uses an actuation scheme based on highly compliant series-elastic actuators that enable torque control. Our aim is to increase StarlETH's repertoire of motions to include faster, more life-like dynamic gaits. To this end we build on the framework described by Coros et al. [8]. The control scheme combines several simple building blocks. An inverted pendulum model computes desired foot fall locations, PD controllers regulate the motions of the legs and virtual forces are used to continuously modulate the position and orientation of the main body. In addition to discussing the changes needed to apply the control scheme to StarlETH, we describe an improved method for distributing virtual forces to the stance legs. We demonstrate the flexibility of the described framework by generating walking and trotting controllers that are robust to pushes and significant, unanticipated variations in the terrain. In addition, we show that our method can produce smooth gait transitions that depend on the walking speed, resulting in increased agility.

II. CONTROL FRAMEWORK

The goal of our control framework is to provide quadruped robots with the ability to move through their environments

while robustly dealing with perturbations due to external pushes or unperceived variations in the terrain. In order to allow the robots to be steered, the desired forward speed v_x^d , lateral speed v_y^d or turning rate ψ^d are treated as high-level parameters that can be modified at any time.

Figure 1 shows the different building blocks of our control system. Our control framework (blue) computes desired joint positions \mathbf{q}^d and torques $\boldsymbol{\tau}^d$ that are passed to the low-level controller (green). The latter considers the dynamics of the actuators and regulates the motor currents I^d . The *motion generator* and *motion controller* modules shown in Fig. 1 illustrate the two core questions addressed by our method: how do we generate appropriate motion objectives for the whole-body system (Section II-B), and how do we best achieve them (Section II-C)? Before we discuss in detail these two components, we first describe the characteristics of the plant.

A. The Plant

Our quadruped robot, StarLETH, has four articulated legs with three actuated degrees-of-freedom (DoF) each: hip abduction/adduction (HAA), hip flexion/extension (HFE), and knee flexion/extension (KFE). The mechanical system therefore has 12 actuated DoFs and 18 DoFs in total. The controller has access to all joint angles $\mathbf{q}_j \in \mathbb{R}^{12}$, joint velocities $\dot{\mathbf{q}}_j$, as well as to the pose $\mathbf{q}_b \in \mathbb{R}^6$ and velocities $\dot{\mathbf{q}}_b$ of the main body, which are estimated by an Extended Kalman filter that fuses IMU data and leg kinematics [12]. The minimal coordinates of the free-floating robot are thus given by $\mathbf{q} = [\mathbf{q}_b, \mathbf{q}_j]^T$. Additionally, pressure sensors in the feet indicate whether the legs are in contact with the ground. By thresholding the pressure readings, a boolean contact flag for each leg ($c_{\text{flag}} \in \{0, 1\}$) is available for the control algorithm.

The control framework described in this paper outputs desired joint angles \mathbf{q}_j^d for the swing legs and desired joint torques $\boldsymbol{\tau}^d$ for the stance legs, as the series-elastic actuators employed by StarLETH enable torque control. The high compliance of the system, however, requires sophisticated low-level torque and position controllers in order to cope with the resulting low control bandwidth. We therefore use the low-level control system described by Hutter et al. [13] to generate desirable motor currents.

B. Motion Generation

The motions of the legs and the main body are described in our framework either in the inertial (world) frame I or in the body frame B that is located at the center of the main body, namely in the center of the HAA joints. The main frame's x-axis is aligned with the robot's heading direction, which we also refer to as the sagittal direction. The vertical direction is collinear with the z-axis of the inertial frame. The y-axis of the main frame denotes the robot's coronal direction.

1) *Terrain*: To plan the locations of the footholds, it is essential to know where the ground is located in the inertial frame. Since the estimated vertical position of the robot can drift, and we restrict ourselves from using any external sensors, we estimate the ground height h_g by filtering the vertical

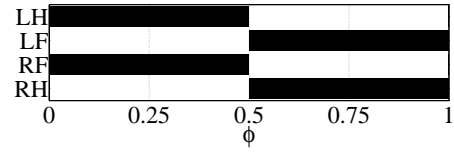


Fig. 2. Gait graph for a walking trot: the black bar defines the stance phase of the left hind (LH), left front (LF), right front (RF), and right hind (RH) leg, respectively.

position of the stance feet, $I r_{F_i,z}$, expressed in the inertial frame:

$$h_g(t) = \sum_{i=0}^{N=4} c_{\text{flag}i} (I r_{F_i,z} \cdot \alpha + h_g(t - \Delta t) \cdot (1 - \alpha)), \quad (1)$$

where $\alpha = 0.2$ is the parameter of a first order filter.

2) *Timing*: Quadruped gaits are to a large extent defined by the foot fall pattern and the duration of the gait cycle T_s . In our implementation, this is controlled by the *Gait Pattern*, which explicitly defines the role of each leg at any moment in time. Legs that are in swing mode need to safely reach the next foothold location in order to ensure that the robot can move at the desired speed or that it can recover balance. In contrast, the legs that are in stance mode must help satisfy the motion objectives of the main body in a coordinated way.

The Gait Pattern defines the sequence of swing and stance modes for each leg with respect to the time-normalized stride phase $\phi \in [0, 1]$, as illustrated in Figure 2. The white areas indicate the fraction of the stride when a leg is in swing phase, which is characterized by the relative timing of the lift-off and touch-down events. The dark areas indicate that a leg is in stance mode. In addition to informing the controller of whether a leg is in swing or stance, the gait pattern is used to estimate the amount of time left before a leg should transition to the next mode. This information is useful as it helps the controller anticipate how the support polygon will change in the near future and plan accordingly.

The stance phase $\phi_{\text{st}} \in [0, 1]$ of a leg indicates the time normalized progress made during the stance mode. The swing phase of a leg, ϕ_{sw} , determines the amount of time left before the next foot touch-down event, and it is set to -1 if the stance phase $\phi_{\text{st}} > 0$. We define the rule used to determine if a leg is in stance mode $\iota_{\text{st}} \in \{0, 1\}$ as:

$$\iota_{\text{st}} = \begin{cases} 1 & \text{if } c_{\text{flag}} \wedge (\phi_{\text{sw}} > 0.9) \\ \phi_{\text{sw}} < 0 & \text{otherwise} \end{cases} \quad (2)$$

The first case employed in the equation above ensures that legs are free to transition to stance mode earlier than predicted in order to support the main body, if early contacts are detected. The swing mode $\iota_{\text{sw}} \in \{0, 1\}$ is defined as $\iota_{\text{sw}} = \neg \iota_{\text{st}}$.

We introduce another variable, the grounded flag $g_{\text{flag}} = \iota_{\text{st}} \wedge c_{\text{flag}} \in \{0, 1\}$, to select the appropriate low-level controller. The flag is only true if the leg is, and should be, in contact with the ground. In this case it is safe to apply torque control at all the joints of the leg, including the knee.

3) *Swing Leg Configuration*: Appropriate foot placement control for the swing legs can provide the robot with the ability to recover balance when it is pushed, or when it encounters

unanticipated variations in the terrain. Our foot placement algorithm currently considers each leg independently of the others. At every control cycle we calculate, for each swing leg, an appropriate foothold position. Subsequently, we plan a trajectory for the foot in order to ensure that the target stepping location is reached safely. This results in desired swing foot positions at every moment in time.

The target foothold location ${}^I\mathbf{r}_F$ is computed relative to the HAA joint ${}^I\mathbf{r}_H$:

$${}^I\mathbf{r}_{HF} = {}^I\mathbf{r}_{HF}^{\text{fb}} + {}^I\mathbf{r}_{HF}^{\text{ff}}, \quad (3)$$

where ${}^I\mathbf{r}_{HF}^{\text{fb}}$ is a feedback term predicted by an inverted pendulum model [14], and ${}^I\mathbf{r}_{HF}^{\text{ff}}$ is a feedforward step length that depends on the robot's desired speed. This formulation is similar to the one described in [15].

We use a slightly modified version of the inverted pendulum prediction in order to compute the feedback component of the stepping location:

$${}^I\mathbf{r}_{HF}^{\text{fb}} = \eta({}^I\mathbf{v}_{ref} - {}^I\mathbf{v}^d) \sqrt{\frac{h}{g}}, \quad (4)$$

where $h = {}^I r_{H,z} - h_g$ is the current height of the hip with respect to the ground, ${}^I\mathbf{v}^d = {}^I v_x^d + {}^I v_y^d$ is the desired velocity, g is the gravitational acceleration, and ${}^I\mathbf{v}_{ref}$ is an estimated reference linear velocity, and η is a scaling parameter that was set to 1.2 for all our experiments. This particular form of the feedback term ensures that, when moving at the desired speed, only the feedforward component of the step location is used. Consequently, only differences between the current speed and the desired speed are taken into account by the feedback component. In practice we noticed that the feedback component of the step can be too large when the robot is mostly rotating about the yaw axis. For this reason, we compute the estimated reference velocity used in the equation above as the average between the leg's hip velocity and that of the body's COM:

$${}^I\mathbf{v}_{ref} = \frac{1}{2}({}^I\mathbf{v}_H + {}^I\mathbf{v}_{CoM}). \quad (5)$$

The feedforward component of the stepping location is computed as half the distance the CoM is expected to travel during the stance duration Δt_{st} that is defined by the Gait Pattern:

$${}^I\mathbf{r}_{HF}^{\text{ff}} = \frac{1}{2}{}^I\mathbf{v}^d \Delta t_{st}. \quad (6)$$

We can optionally add an additional offset, ${}^B\mathbf{r}_{HF}^d$, to the feedforward stepping location in order to control the width of the steps that are taken. This is particularly useful for slower gaits such as the static walk.

The stepping offset ${}^I\mathbf{r}_{HF}$ constitutes the final desired location for foot placement. However, we need to provide the robot with a continuous trajectory that ensures that the final foot location can be reached safely. We therefore linearly interpolate between the initial location of the foot at the beginning of the swing phase, and this final target location.

To provide enough ground clearance for the foot, we use a pre-defined height trajectory that varies as a function of the swing phase, as shown in Fig. 3a. This trajectory is defined by a spline, and all values are relative to the estimated ground height h_g .

4) *Stance Leg Configuration*: In case a leg is in stance mode according to the Gait Pattern, but loses contact with the ground ($g_{\text{flag}} = 1$), we compute a desired foot target that is 1cm lower than the leg's current position, in order to regain contact with the ground as soon as possible. Otherwise, because there are no kinematic redundancies in the mechanical design, we do not need to actively control the pose of the stance legs.

5) *Main Body Configuration*: The pose \mathbf{q}_b and velocity $\dot{\mathbf{q}}_b$ of the main body need to be controlled in order to increase robustness, i.e. prevent the robot from tumbling over, and to meet the desired velocity commands. By default, the desired orientation of the main body is defined by zero roll and pitch angles, whereas the yaw angle is unconstrained. The desired height of the body relative to the estimated ground height, h_H , is specified by a spline as a function of the stride phase, and it can be used, for instance, to propel the body upwards at the right moment in time in anticipation of a flight phase. The desired position of the body along the sagittal and coronal directions is computed relative to the positions of the feet:

$${}^I\mathbf{r}_B^d = \frac{\sum_{i=1}^N w_i(\phi) {}^I\mathbf{r}_{Fi}}{\sum_{i=1}^N w_i(\phi)}, \quad (7)$$

where the leg weights w_i depend on the stride phase ϕ as illustrated in Fig. 3b. We compute the desired position based not only on the grounded legs, but also based on the swing legs, in order to get smooth trajectories for the desired position of the body. With the strategy we implemented, as a grounded leg approaches the end of the stance phase ($\phi_{st} = \phi_{st,0}$), the body can start shifting away from it. Similarly, the body starts shifting towards a swing leg, as it reaches the end of the swing phase ($\phi_{sw} = \phi_{sw,0}$) and prepares for landing. This anticipatory behavior is flexible enough to control traditional static gaits, and it allows us to also implement dynamic gaits that are increasingly more agile. The minimal weight w_{\min} depends on the gait and is found experimentally.

The generalized desired position of the main body is given by $\mathbf{q}_b^d = ({}^I r_{B,x}^d, {}^I r_{B,y}^d, h_g + h_H(\phi), 0, 0, 0)^T$, while its desired velocity is $\dot{\mathbf{q}}_b^d = (v_x^d, v_y^d, 0, 0, 0, \dot{\psi}^d)^T$.

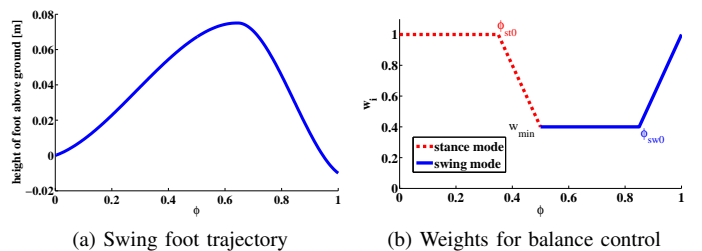


Fig. 3. Most of the motion characteristics are described with respect to the stride phase ϕ .

C. Motion Control

We use low level position controllers in order to get fast and precise tracking of the swing leg joint trajectories [13]. For the legs that are in stance mode, we make use of virtual force control, as it is an intuitive and effective method. However, due to the particular mechanical setup of the knee joint, we cannot apply torque control to the knee joints of the stance legs unless the knee spring is under tension. In practice, we detect ground contacts, or lack thereof, at a fast enough rate to employ a hybrid control approach, using torque control for the stance legs that are in contact with the ground, and position control otherwise.

1) *Position Control*: We use position control whenever a leg is not, or should not be ($g_{\text{flag}} = 0$), in contact with the ground. The desired joint angles \mathbf{q}_j^d are obtained from the desired foot positions through inverse kinematics, and are then passed directly to the low level controller.

2) *Torque Control*: The joint torques that need to be applied through the stance legs are computed in three steps. We first compute virtual forces and torques that should ideally act on the main body in order to control the robot's posture. These are then optimally distributed to the stance legs given the current kinematic configuration of the robot. Lastly, we map the virtual leg forces \mathbf{F}_{leg} to the joint torques by applying Jacobian transpose control: $\boldsymbol{\tau} = \mathbf{J}^T \mathbf{F}_{\text{leg}}$.

The desired forces and torques that should act on the main body are computed based on the desired pose \mathbf{q}_b^d , the current pose \mathbf{q}_b and their derivatives, as:

$$\begin{bmatrix} {}_B\mathbf{F}_B^d \\ {}_B\mathbf{T}_B^d \end{bmatrix} = \mathbf{k}_p(\mathbf{q}_b^d - \mathbf{q}_b) + \mathbf{k}_d(\dot{\mathbf{q}}_b^d - \dot{\mathbf{q}}_b) + \mathbf{k}_{ff} \begin{pmatrix} v_x^d \\ v_y^d \\ mg \\ 0 \\ 0 \\ \psi^d \end{pmatrix} \quad (8)$$

where \mathbf{k}_p , \mathbf{k}_d , and \mathbf{k}_{ff} are the proportional, derivative and feed-forward gains, respectively and m is the total mass of the robot. The feed-forward gains improve tracking the desired velocities and compensate for gravity.

The desired net virtual force ${}_B\mathbf{F}_B^d$ and torque ${}_B\mathbf{T}_B^d$ that should be applied to the main body are bounded before being distributed to the stance legs, in order to ensure that the robot does not apply excessively large forces through the stance legs. In the framework described in [8], ${}_B\mathbf{F}_B^d$ and ${}_B\mathbf{T}_B^d$ are equally distributed to the stance legs. This strategy did not work for our static walking gait. Instead, at each control step, we solve a convex optimization problem with linear constraints in order to compute desirable contact and friction forces to be applied through the stance feet. More formally, the problem formulation is as follows:

$$\text{minimize} \quad (\mathbf{A}\mathbf{x} - \mathbf{b})^T \mathbf{S}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \mathbf{x}^T \mathbf{W}\mathbf{x} \quad (9)$$

$$\text{subject to} \quad F_{\text{leg},i}^n \geq F_{\text{min}}^n, \quad (10)$$

$$-\mu F_{\text{leg},i}^n \leq F_{\text{leg},i}^t \leq \mu F_{\text{leg},i}^n \quad (11)$$

where $\mathbf{x} = [\mathbf{F}_{\text{leg},0}^T, \dots, \mathbf{F}_{\text{leg},i}^T, \dots, \mathbf{F}_{\text{leg},m}^T]^T$, $\mathbf{F}_{\text{leg},i}^T$ represents the net force to be applied through the i^{th} stance leg and m is the number of legs that are and should be grounded ($g_{\text{flag}} = 1$).

In order to ensure that the forces applied through the stance legs result in a net force and torque that are as close as possible to the desired values, we compute \mathbf{A} and \mathbf{b} using:

$$\underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \\ \mathbf{r}_0 \times & \mathbf{r}_1 \times & \dots & \mathbf{r}_m \times \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \mathbf{F}_{\text{leg},0} \\ \mathbf{F}_{\text{leg},1} \\ \vdots \\ \mathbf{F}_{\text{leg},m} \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} \mathbf{F}_B^d \\ \mathbf{T}_B^d \end{pmatrix}}_{\mathbf{b}}, \quad (12)$$

where \mathbf{r}_i is the vector between the CoM and the location of the foot of stance leg i . The weighting matrix \mathbf{S} trades off the degree to which we want to match the net resulting torque over the net resulting force, and the term $\mathbf{x}^T \mathbf{W}\mathbf{x}$ acts as a regularizer that discourages the use of large virtual forces.

The constraints applied ensure that the normal component of the force applied through each leg, $F_{\text{leg},i}^n$, is strictly positive (no pulling on the ground). In practice we found that requiring a minimal force $F_{\text{min}}^n = 2\text{N}$ to always be applied results in fewer instances where the feet slip. We also restrict the tangential component $F_{\text{leg},i}^t$ to remain within an approximate friction cone defined by the assumed friction coefficient $\mu = 0.8$ in order to avoid slipping.

D. Gait Transitions




Gaits are mainly characterized by the gait pattern and the stride duration, but several parameters in our control framework have to be adjusted specifically for each gait in order to increase performance. Fortunately, we have observed that smooth transitions between gaits can be generated by linearly interpolating the individual parameter sets. As long as the gait patterns are compatible (there is no smooth transition between trot and gallop, but there is one between walking to trotting, for instance), this approach seems to work well and does not require additional parameter tuning. However, it is likely that the resulting transitions may be suboptimal. We define a time horizon for the interpolation procedure, and the gait transitions are either initiated manually by an operator, or as a function of the desired speed.

III. RESULTS

Before conducting any experiments on StarLETH, we verified the control framework in simulation. Here we only discuss the results obtained by running the control strategy on the physical robot. Our results are best seen in the accompanying video. For more information about the simulation environment and the software package we refer the interested reader to Hutter et al. [11].

StarLETH was able to move freely in 3D during all our experiments, and was not aided by any support structures. A static walk, a walking trot, and a running trot (with flight phase) were successfully implemented on StarLETH. The walking trot reached a top speed of 0.7m/s on a treadmill

TABLE I
PARAMETER SETS FOR DIFFERENT GAITS

Parameter	Symbol	Static Walk	Walking Trot	Running Trot
gait graph				
stride duration	$T_s [s]$	1.5	0.8	0.7
min. leg weight for support polygon	w_{\min}	0.35	0.15	0.15
start of increasing the weight of a swing leg for support polygon	$\phi_{sw,0}$	0.7	0.7	0.7
start of decreasing the weight of the stance leg for support polygon	$\phi_{st,0}$	0.7	0.7	0.7
default left front swing leg offset	${}^B r_{HF}^d [m]$	$[0, -0.01, 0]^T$	$[0, 0, 0]^T$	$[0, 0, 0]^T$
default left hind swing leg offset	${}^B r_{HF}^d [m]$	$[0, 0.14, 0]^T$	$[0, 0, 0]^T$	$[0, 0, 0]^T$
height of middle of hip AA joints	$h_H [m]$	0.39	0.44	0.44
virt. force proportional gain	\mathbf{k}_p	$[500, 640, 600, 400, 200, 0]^T$	$[0, 640, 600, 400, 200, 0]^T$	$[0, 640, 2600, 400, 200, 0]^T$
virt. force derivative gain	\mathbf{k}_d	$[150, 100, 120, 6, 9, 0]^T$	$[150, 100, 120, 6, 9, 0]^T$	$[90, 60, 120, 6, 9, 0]^T$
virt. force feed-forward gain	\mathbf{k}_{ff}	$[25, 0, 1, 0, 0, 0]^T$	$[60, 0, 1, 0, 0, 0]^T$	$[25, 0, 1, 0, 0, 0]^T$
weights for matching the des. virt. forces	S	$\text{diag}(1, 1, 1, 10, 10, 5)$	$\text{diag}(1, 1, 0.2, 20, 20, 5)$	$\text{diag}(1, 1, 0.2, 20, 20, 5)$
weights for reducing joint torques	W	$\text{diag}(0.00001 \dots)$	$\text{diag}(0.00001 \dots)$	$\text{diag}(0.00001 \dots)$

whose speed was set to match that of the robot, as measured by a motion capture system.

A. Parameter Sets

We tuned the initial gains of the virtual force controller while perturbing the robot as it tried to stand in place. We then adjusted the parameters for the different gaits while the robot was walking or trotting. We found this process to be intuitive, because a large range of parameters result in successful motions, and the parameters are largely orthogonal as they affect different aspects of the motion objectives or motion control components. The parameters we used for the static walk, walking trot, and running trot are summarized in Table I.

B. Robustness

The robustness of the control system was examined by asking the robot to walk and trot on flat ground, while introducing unanticipated obstacles up to 5cm high (an eighth of the leg length) as shown in Fig. 5. In addition, we tested the ability of the robot to recover from external pushes. While the duration of the push, the current phase in the locomotion cycle and the push direction can affect the ability of the robot to reject perturbations, we noticed that significant pushes are generally handled well (as shown in the supplementary video). The foot placement strategy, in conjunction with the appropriate distribution of virtual forces to the stance legs allowed the robot to successfully recover from various such scenarios. When the robot failed to recover balance, this was typically due to the HAA joints reaching their joint limits while the legs were in stance mode.

Figure 4 presents some relevant data from one of the push experiments we performed. As seen in the supplementary video, StarLETH was pushed in the sagittal direction for a duration of roughly 0.2s (indicated by the gray area). The first sub-plot in Fig. 4 shows the sagittal position. The robot moves forward during the push and soon thereafter steps in order to recover balance. The second plot shows the coronal position, where a slight lateral drift is visualized. The following three

plots show the net virtual forces for the main body in the sagittal and coronal directions, as well as the net torque about the y-axis of the robot. The solid lines illustrate the desired virtual forces, whereas the dashed lines show the sum of the contributions of the distributed leg forces. As can be seen, the force distribution favors matching the desired torque over matching the desired forces, as indicated by the input weighting matrix S . When all four legs are in contact with ground, the net distributed forces begin to match both the desired forces and the desired torques, as there are enough degrees of freedom in the system. The influence of the unilateral contact constraints can also be observed in these plots. When only two legs are in contact with the ground, the errors in the distributed coronal force result in the drift observed in the second plot. The last plot shows the measured (solid) and desired (dashed) joint torque in the knee joint. The swing phase can be clearly identified by the zero joint torque.

C. Gait Transitions

StarLETH can smoothly transition from the static walk to the walking trot if we linearly interpolate between the parameter sets shown in Table I. The duration of the interpolation can be chosen somewhat arbitrarily, but for the results we showed here we used a time period of 3s. The transition from the trot to the walk takes place over 0.5s. We noticed that the transitions are robust with respect to the exact stride phase when they are initiated, and we therefore do not require them to start at a particular point in the locomotion cycle. To transition between the walking trot and the flying trot we similarly interpolate the parameter sets of the two gaits.

IV. CONCLUSION

The control framework described by Coros et al. [8] was extended to enable our quadruped robot to perform a static walk, a walking trot and a running trot. In addition to detailing the various changes needed to apply this control framework to a real robot, we employed a new force distribution method, without which the robot was unable to walk.

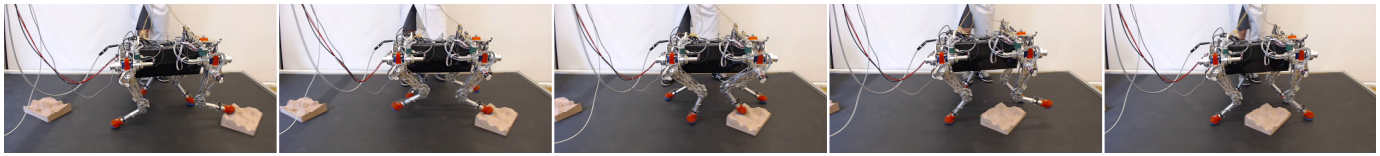


Fig. 5. StarLETH performs a walking trot while dealing with unperceived obstacles.

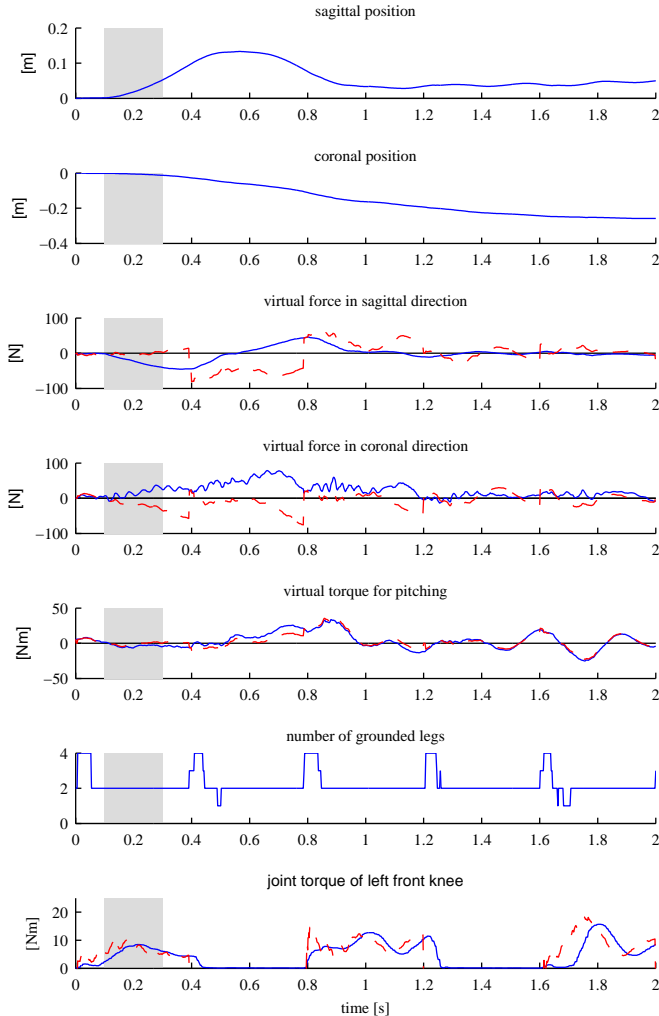


Fig. 4. Experimental results of a push that was applied in sagittal direction during 0.2s as indicated by the grey area.

The main benefits of the framework that we used are that it is highly modular, that it allows the various parameters to be tuned in an intuitive way, and that it results in locomotion controllers that are robust to pushes and unexpected variations in the terrain. As shown in simulation, the parameter space of this control framework is rich enough to also describe other gaits, such as a pace, bound or gallop [8]. In the future we plan on further extending StarLETH's repertoire of motions by creating controllers and transitions for this new set of gaits.

We have not yet performed a quantitative evaluation of the performance and robustness of the control system, and this will be part of future investigations. The clean separation of the motion generator and the motion controller modules

will enable us to also compare different control strategies. For instance, different models can be plugged in for the foot placement component, or the force distribution method could be replaced by an operational space approach [16] in order to test the relative merits of the different building blocks we use. Last but not least, we plan to investigate a systematic way of finding optimal parameter sets on the real system.

REFERENCES

- [1] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.
- [2] P. González-de Santos, E. Garcia, and J. Estremera, *Quadrupedal locomotion: an introduction to the control of four-legged robots*. Springer, 2006.
- [3] M. Hildebrand, "The Quadrupedal Gaits of Vertebrates," *BioScience*, vol. 39, no. 11, pp. 766–775, Dec. 1989.
- [4] M. P. Murphy, A. Saunders, C. Moreira, A. A. Rizzi, and M. Raibert, "The little dog robot," *The International Journal of Robotics Research*, 2010.
- [5] M. Raibert, "Bigdog, the rough-terrain quadruped robot," in *Proceedings of the 17th IFAC World Congress*, M. J. Chung, Ed., vol. 17, no. 1, 2008.
- [6] C. Semini, N. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. Caldwell, "Design of hyq—a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.
- [7] J. B. I. Havoutis, C. Semini and D. Caldwell, "Progress in quadrupedal trotting with active compliance," *Dynamic Walking*, 2012.
- [8] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne, "Locomotion skills for simulated quadrupeds," *ACM Transactions on Graphics*, vol. 30, no. 4, 2011.
- [9] D. Krasny and D. Orin, "Evolution of a 3d gallop in a quadrupedal model with biological characteristics," *Journal of Intelligent and Robotic Systems*, vol. 60, pp. 59–82, 2010.
- [10] M. H. Raibert, "Symmetry in running," *Science*, vol. 231, no. 4743, pp. 1292–1294, 1986.
- [11] M. Hutter, C. Gehring, M. Bloesch, M. Hoepflinger, C. Remy, and R. Siegwart, "StarLETH: a compliant quadrupedal robot for fast, efficient, and versatile locomotion," in *Proc. of the International Conference on Climbing and Walking Robots (CLAWAR)*, 2012.
- [12] M. Bloesch, M. Hutter, M. H. Hoepflinger, C. D. Remy, C. Gehring, and R. Siegwart, "State estimation for legged robots - consistent fusion of leg kinematics and IMU," *Proceedings of Robotics: Science and Systems*, 2012.
- [13] M. Hutter, C. D. Remy, M. H. Hoepflinger, and R. Siegwart, "High compliant series elastic actuation for the robotic leg ScarLETH," in *Int. Conference on Climbing and Walking Robots (CLAWAR)*, 2011.
- [14] J. E. Pratt and R. Tedrake, "Velocity-based stability margins for fast bipedal walking," *Fast Motions in Biomechanics and Robotics*, vol. 340, pp. 1–27, 2006.
- [15] M. H. Raibert, *Legged Robot that Balance*. MIT Press, 1986.
- [16] M. Hutter, M. Hoepflinger, C. Gehring, M. Bloesch, C. D. Remy, and R. Siegwart, "Hybrid operational space control for compliant legged systems," in *Proceedings of Robotics: Science and Systems*, 2012.