# Bounding on Rough Terrain with the LittleDog Robot

Alexander Shkolnik, Michael Levashov, Ian R. Manchester and Russ Tedrake

*The International Journal of Robotics Research* published online 7 December 2010
DOI: 10.1177/0278364910388315

The International Journal of
Robotics Research
00(000) 1–24
© The Author(s) 2010
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364910388315
ijr.sagepub.com

# Bounding on Rough Terrain with the LittleDog Robot

Alexander Shkolnik, Michael Levashov, Ian R. Manchester and Russ Tedrake

**Abstract**

*A motion planning algorithm is described for bounding over rough terrain with the LittleDog robot. Unlike walking gaits, bounding is highly dynamic and cannot be planned with quasi-steady approximations. LittleDog is modeled as a planar five-link system, with a 16-dimensional state space; computing a plan over rough terrain in this high-dimensional state space that respects the kinodynamic constraints due to underactuation and motor limits is extremely challenging. Rapidly Exploring Random Trees (RRTs) are known for fast kinematic path planning in high-dimensional configuration spaces in the presence of obstacles, but search efficiency degrades rapidly with the addition of challenging dynamics. A computationally tractable planner for bounding was developed by modifying the RRT algorithm by using: (1) motion primitives to reduce the dimensionality of the problem; (2) Reachability Guidance, which dynamically changes the sampling distribution and distance metric to address differential constraints and discontinuous motion primitive dynamics; and (3) sampling with a Voronoi bias in a lower-dimensional "task space" for bounding. Short trajectories were demonstrated to work on the robot, however open-loop bounding is inherently unstable. A feedback controller based on transverse linearization was implemented, and shown in simulation to stabilize perturbations in the presence of noise and time delays.*

**Keywords**

Legged locomotion, rough terrain, bounding, motion planning, LittleDog, RRT, reachability-guided RRT, transverse linearization

## 1. Introduction

While many successful approaches to dynamic legged locomotion exist, we do not yet have approaches which are general and flexible enough to cope with the incredible variety of terrain traversed by animals. Progress in motion planning algorithms has enabled general and flexible solutions for slowly moving robots, but we believe that in order to quickly and efficiently traverse very difficult terrain, extending these algorithms to dynamic gaits is essential.

In this work we present progress towards achieving agile locomotion over rough terrain using the LittleDog robot. LittleDog (Figure 1) is a small, 3-kg position-controlled quadruped robot with point feet and was developed by Boston Dynamics under the DARPA Learning Locomotion program. The program ran over several phases from 2006 to 2009, and challenged six teams from universities around the United States to compete in developing algorithms that enable LittleDog to navigate known, uneven terrain, as quickly as possible. The program was very successful, with many teams demonstrating robust planning and locomotion over quite challenging terrain (e.g. Pongas et al. 2007; Rebula et al. 2007; Kolter et al. 2008; Zucker 2009), with an emphasis on walking gaits, and some short stereotyped dynamic maneuvers that relied on an intermittent existence of a support polygon to regain control and simplify planning (Byl et al. 2008). In this paper, we present a method for generating a continuous dynamic bounding gait over rough terrain.

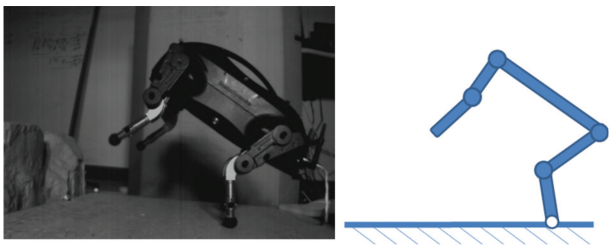
**Fig. 1.** LittleDog robot, and a corresponding five-link planar model.

Computer Science and Artificial Intelligence Lab, MIT, Cambridge, MA, USA

**Corresponding author:**
Alexander Shkolnik, Computer Science and Artificial Intelligence Lab, MIT, 32 Vassar Street, Cambridge, MA 02139, USA.
Email: shkolnik@csail.mit.edu

Achieving bounding on LittleDog is difficult for a number of reasons. First, the robot is mechanically limited: high-gear ratio transmissions generally provide sufficient torque but severely limit joint velocities and complicate any attempts at direct torque control. Second, a stiff frame complicates impact modeling and provides essentially no opportunity for energy storage. Finally, and more generally, the robot is underactuated, with the dynamics of the unactuated joints resulting in a complicated dynamical relationship between the actuated joints and the interactions with the ground. These effects are dominant enough that they must be considered during the planning phase.

In this work, we propose a modified form of the Rapidly Exploring Random Tree (RRT) (LaValle and Kuffner 2001) planning framework to quickly find feasible motion plans for bounding over rough terrain. The principal advantage of the RRT is that it respects the kinematic and dynamic constraints which exist in the system, however for high-dimensional robots the planning can be prohibitively slow. We highlight new sampling approaches that improve the RRT efficiency. The dimensionality of the system is addressed by biasing the search in a low-dimensional task space. A second strategy uses reachability guidance as a heuristic to encourage the RRT to explore in directions that are most likely to successfully expand the tree into previously unexplored regions of state space. This allows the RRT to incorporate smooth motion primitives, and quickly find plans despite challenging differential constraints introduced by the robot's underactuated dynamics. This planner operates on a carefully designed model of the robot dynamics which includes the subtleties of motor saturations and ground interactions.

Bounding motions over rough terrain are typically not stable in open loop: small disturbances away from the nominal trajectory or inaccuracies in physical parameters used for planning can cause the robot to fall over. To achieve reliable bounding it is essential to design a feedback controller to robustly stabilize the planned motion. This stabilization problem is challenging due to the severe underactuation of the robot and the highly dynamic nature of the planned trajectories. We implemented a feedback controller based on the method of *transverse linearization* which has recently emerged as an enabling technology for this type of control problem (Hauser and Chung 1994; Shiriaev et al. 2008; Manchester et al. 2009; Manchester 2010).

The remainder of this paper is organized as follows: in Section 2 we begin by reviewing background information, including alternative approaches to achieve legged locomotion over rough terrain with particular attention given to motion planning approaches. In Section 3 we present a comprehensive physics-based model of the LittleDog robot and the estimation of its parameters from experiments. Section 4 discusses motion planning for bounding, with a detailed description of the technical approach and experimental results. In Section 5, we describe the feedback control design and show some simulation and experimental results. Section 6 concludes the paper, and discusses remaining open problems.

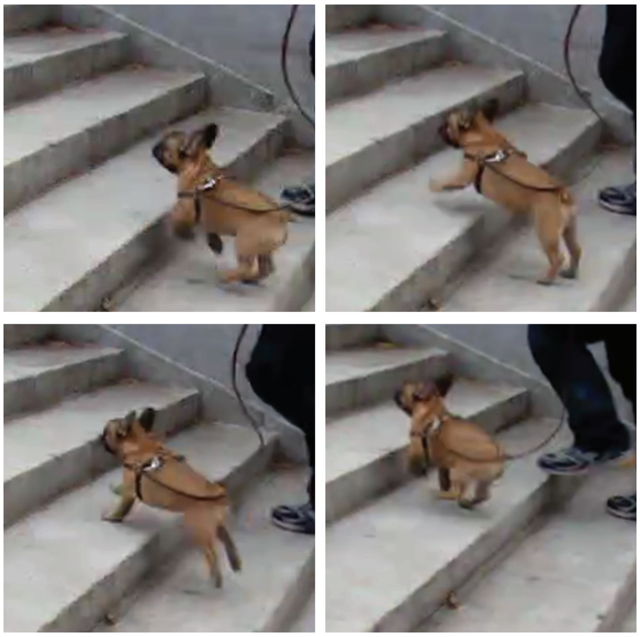
**Fig. 2.** Dog bounding up stairs. Images from video available at http://www.flickr.com/photos/istolethetv/3321159490/

## 2. Background

The problem of fast locomotion over rough terrain has been an active research topic in robotics, beginning with the seminal work by Raibert in the 1980s (Raibert 1986; Raibert et al. 1986). The research can be roughly divided into two categories. The first category uses knowledge of the robot and environment within a motion planning framework. This approach is capable of versatile behavior over rough terrain, but motion plans tend to be slow and conservative. The other category is characterized by a limit-cycle approach, which is usually blind to the environment. In this approach, more dynamic motions may be considered, but typically over only a limited range of behavior. In this section we review these approaches in turn.

### 2.1. Planning Approaches

Planning algorithms have made significant headway in recent years. These methods are particularly well developed for kinematic path planning in configuration space, focusing on maneuvers requiring dexterity, obstacle avoidance, and static stability. Sampling-based methods such as the RRT are very effective in planning in high-dimensional humanoid configuration spaces. The RRT has been used to plan walking and grasping trajectories amidst obstacles by searching for a collision-free path in configuration space, while constraining configurations to those that are statically stable (Kuffner et al. 2002, 2003). The robot is statically stable when the center of mass (COM) is directly above the support polygon, therefore guaranteeing that the robot will not roll over as long as the motion is executed slowly enough. After finding a statically feasible trajectory of configurations (initially ignoring velocity), the trajectory is locally optimized for speed and smoothness, while maintaining the constraint that at least one foot remains flat on the ground at all times. This approach has been extended to account for moving obstacles and demonstrated on the Honda Asimo (Chestnutt et al. 2005). An alternative approach is to first generate a walking pattern while ignoring obstacles and collisions, and then use random sampling to modify the gait to avoid obstacles while verifying constraints to ensure the robot does not fall (Harada et al. 2007). Current methods are adept at planning in high-dimensional configuration spaces, but typically only for limited dynamic motions. Sampling-based planning algorithms are in general not well suited for planning fast dynamic motions, which are governed largely by underactuated dynamics.

The use of static stability for planning allows one to ignore velocities, which halves the size of the state space, and constrains the system to be fully actuated, which greatly simplifies the planning problem. Statically stable motions are, however, inherently conservative (technically a robot is truly statically stable only when it is not moving). This constraint can be relaxed by using a dynamic stability criteria (see Pratt and Tedrake (2005) for review of various metrics). These metrics can be used either for gait generation by the motion planner, or as part of a feedback control strategy. One popular stability metric requires the center of pressure, or the Zero Moment Point (ZMP), to be within the support polygon defined by the convex hull of the feet contacts on the ground. While the ZMP is regulated to remain within the support polygon, the robot is guaranteed not to roll over any edge of the support polygon. In this case, the remaining degrees of freedom can be controlled as if the system is fully actuated using standard feedback control techniques applied to fully actuated systems. Such approaches have been successfully demonstrated for gait generation and execution on humanoid platforms such as the Honda Asimo (Sakagami et al. 2002; Hirose and Ogawa 2007), and the HRP series of walking robots (Kaneko et al. 2004). Lower-dimensional "lumped" models of the robot can be used to simplify the differential equations that define ZMP. In Kajita et al. (2003), the HRP-2 robot was modeled as a cart (rolling point mass) on top of a table (with a small base that should not roll over). Preview control was then applied to generate a COM trajectory which regulates the desired ZMP position. Note that although the ZMP is only defined on flat terrain, some extensions can be applied to extend the idea to 3D, including using hand contacts for stability, for example by considering the Contact Wrench Sum (CWS) (Sugihara 2004).

The works described so far were mostly demonstrated on relatively flat terrain. Other quasi-static planning approaches were applied to enable climbing behavior and walking over varied terrain with the HRP-2 humanoid robot (Hauser et al. 2008; Hauser 2008). In that work, contact points and equilibrium (static) stances, acting as way points, were chosen by using a Probabilistic Road Map (Kavraki et al. 1996), an alternative sampling-based planning strategy. The planner searched for paths through potentially feasible footholds and stances, while taking into account contact and equilibrium constraints to ensure that the robot maintains a foot or hand hold, and does not slip. Motion primitives were then used to find quasi-static local motion plans between stances that maintain the non-slip constraints. With a similar goal of traversing very rough terrain with legged robots, the DARPA Learning Locomotion project, utilizing the LittleDog robot, has pushed the envelope of walking control by using careful foot placement. Much of the work developed in this program has combined path planning and motion primitives to enable crawling gaits on rough terrain (e.g. Pongas et al. 2007; Rebula et al. 2007; Kolter et al. 2008; Ratliff et al. 2009). Similar to the approaches of the other teams, during the first two phases of the program, MIT utilized heuristics over the terrain to assign a cost to potential footholds. A* search was then used to find a trajectory of feasible stances over the lowest cost footholds, and a ZMP-based body motion and swing-foot motion planner was used to generate trajectories to walk over rough terrain.

Use of stability metrics such as ZMP have advanced the state of the art in planning gaits over rough terrain, however these stability constraints result in very conservative dynamic trajectories, for example by requiring that at least one foot is always flat on the ground. Because ZMP-based control systems do not reason about underactuated dynamics, the humanoid robots do not perform well when walking on very rough or unmodeled terrain, cannot move nearly as quickly as humans, and use dramatically more energy (appropriately scaled) than a human (Collins et al. 2005). Animals do not constrain themselves to such a regime of operation. Much more agile behavior takes place precisely when the system is operating in an underactuated regime, for example during an aerial phase, or while rolling the foot over the toe while walking. In such regimes, there is no support polygon, so the legged robot is essentially falling and catching itself. Furthermore, underactuated dynamics might otherwise be exploited: for example, a humanoid robot can walk faster and with longer strides by rolling the foot over the toe.

## 2.2. Limit-cycle Approach

Somewhat orthogonal to the planning approaches, a significant body of research focuses on limit-cycle analysis for walking. Tools developed for limit-cycle analysis allow one to characterize the behavior of a particular gait, typically on flat terrain. Stable limit-cycle locomotion can be achieved by using compliant or mechanically clever designs that enable passive stability using open-loop gaits (e.g. Collins et al. 2005), or otherwise through the use of reflexive control algorithms that tend to react to terrain. Recent applications of these strategies, for example on the Rhex robot (Altendorfer et al. 2001) and BigDog (Raibert et al. 2008), have produced impressive results, but these systems do not take into account knowledge about upcoming terrain.

Feedback control of underactuated "dynamic walking" bipeds has recently been approached using a variety of control methods, including virtual holonomic constraints (Chevallereau et al. 2003; Westervelt et al. 2003, 2007) with which impressive results have been demonstrated for a single limit-cycle gait over flat terrain. In this paper, we use an alternative method based on the combination of transverse linearization and time-varying linear control techniques (Shiriaev et al. 2008; Manchester et al. 2009; Manchester 2010). This allows one to stabilize more general motions, however a nominal trajectory is required in advance, so this feedback controller must be paired with a motion planning algorithm which takes into account information about the environment.

### *2.3. Dynamic Maneuvers*

In order to use sample-based planning for a highly dynamic, underactuated robot, the search must take place in the complete state space, as velocities play an important role in the dynamics. This effectively doubles the dimension of the search. Furthermore, when there are underactuated dynamics, the robot cannot accelerate in arbitrary directions, and therefore can only move in state space in very limited directions. This makes straightforward application of sample-based planning extremely challenging for these types of systems. In the second phase of the LittleDog program, our team began to integrate dynamic lunging, to move two feet at a time (Byl et al. 2008; Byl and Tedrake 2009), into the otherwise quasi-static motion plans to achieve fast locomotion over rough terrain. This paper describes the MIT team approach in the third (final) phase of the project. We show that careful foot placement can be combined with highly dynamic model-based motion planning and feedback control to achieve continuous bounding over very rough terrain.

## 3. LittleDog Model

An essential component of any model-based planning approach is a sufficiently accurate identification of the system dynamics. Obtaining an accurate dynamic model for LittleDog is challenging owing to subtleties in the ground interactions and the dominant effects of motor saturations and transmission dynamics. These effects are more pronounced in bounding gaits than in walking gaits, due to the increased magnitude of ground reaction forces at impact and the perpetual saturations of the motor; as a result, we required a more detailed model. In this section, we describe our system identification procedure and results.

The LittleDog robot has 12 actuators (two in each hip, one in each knee) and a total of 22 essential degrees of freedom (six for the body, three rotational joints in each leg, and one prismatic spring in each leg). By assuming that the leg springs are over-damped, yielding first-order dynamics, we arrive at a 40-dimensional state space ($18 \times 2 + 4$). However, to keep the model as simple (low-dimensional) as possible, we approximate the dynamics of the robot using a planar five-link serial rigid-body chain model, with revolute joints connecting the links, and a free base joint, as shown in Figure 3. The planar model assumes that the back legs move together as one and the front legs move together as one (see Figure 1). Each leg has a single hip joint, connecting the leg to the main body, and a knee joint. The foot of the real robot is a rubber-coated ball that connects to the shin through a small spring (force sensor), which is constrained to move along the axis of the shin. The spring is stiff, heavily damped, and has a limited travel range, so it is not considered when computing the kinematics of the robot, but is important for computing the ground forces. In addition, to reduce the state space, only the length of the shin spring is considered. This topic is discussed in detail as part of the ground contact model.

The model's seven-dimensional configuration space, $\mathcal{C} = \mathbb{R}^2 \times \mathbb{T}^5$, consists of the planar position of the back foot $(x, y)$, the pitch angle $\omega$, and the 4 actuated joint angles $q_1, \ldots, q_4$. The full state of the robot, $x = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{l}] \in \mathcal{X}$, has 16 dimensions and consists of the robot configuration, the corresponding velocities, and the two prismatic shin-spring lengths, $\mathbf{l} = [l_1, l_2]$, one for each foot. The control command, $\mathbf{u}$, specifies reference angles for the four actuated joints. The robot receives joint commands at 100 Hz and then applies an internal PD controller at 500 Hz. For simulation, planning and control purposes, the dynamics are defined as

$$\mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n]), \tag{1}$$

where $\mathbf{x}[n+1]$ is the state at $t[n+1]$, $\mathbf{x}[n]$ is the state at $t[n]$, and $\mathbf{u}[n]$ is the actuated joint position command applied during the time interval between $t[n]$ and $t[n+1]$. We sometimes refer to the control time step, $\Delta T = t[n+1] - t[n] = 0.01$ seconds. A fixed-step fourth-order Runge–Kutta integration of the continuous Euler–Lagrange dynamics model is used to compute the state update.

A self-contained motor model is used to describe the movement of the actuated joints. Motions of these joints are prescribed in the five-link system, so that as the dynamics are integrated forward, joint torques are back-computed, and the joint trajectory specified by the model is exactly followed. This model is also constrained so that actuated joints respect bounds placed on angle limits, actuator velocity limits, and actuator torque limits. In addition, forces computed from a ground contact model are applied to the five-link chain when the feet are in contact with the ground. The motor model and ground contact forces are described in more detail below. The actuated joints are relatively stiff, so the model is most important for predicting the motion of

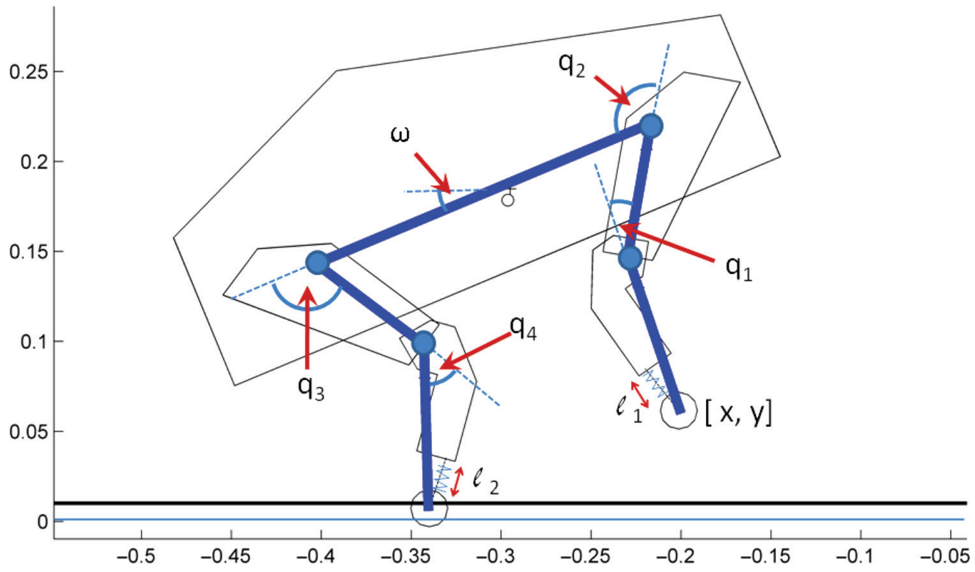

**Fig. 3.** LittleDog model. The state space is $\mathcal{X} = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{l}]$, where $q = [x, y, \omega, q_1, q_2, q_3, q_4]$, and $\mathbf{l} = [l_1, l_2]$ are feet spring lengths used in the ground contact model. The diagram also illustrates the geometric shape of the limbs and body, information used for collision detection during planning.

the unactuated degrees of freedom of the system, in particular the pitch angle, as well as the horizontal position of the robot.

## 3.1. Motor Model

The motors on LittleDog have gear ratios of approximately 70 : 1. Because of the high gear ratio, the internal second-order dynamics of the individual motors dominate in most cases, and the rigid-body dynamics of a given joint, as well as effects of inertial coupling and external forces on the robot can be neglected. The combination of the motor internal dynamics with the PD controller with fixed PD gains can be accurately modeled as a linear second-order system:

$$\ddot{q}_i = -b\dot{q}_i + k(u_i - q_i), \qquad (2)$$

where $\ddot{q}_i$ is the acceleration applied to the $i$th joint, given the state variables $[q_i, \dot{q}_i]$ and the desired position $u_i$. To account for the physical limitations of actual motors, the model includes hard saturations on the velocity and acceleration of the joints. The velocity limits, in particular, have a large effect on the joint dynamics.

Each of the four actuated joints is assumed to be controlled by a single motor, with both of the knee joints having one pair of identical motors, and the hip joints having a different pair of identical motors (the real robot has different mechanics in the hip versus the knee). Owing to this, two separate motor parameter sets: $\{b, k, v_{\text{lim}}, a_{\text{lim}}\}$ are used, one for the knees, and one for the hips.

Figure 4 shows a typical fit of the motor model to real trajectories. The fits are consistent across the different joints of the robot and across different LittleDog robots, but depend on the gains of the PD controller at each of the joints. As seen from the figure, the motor model does well in tracking the actual joint position and velocity. Under large dynamic loads, such as when the hip is lifting and accelerating the whole robot body at the beginning of a bound, the model might slightly lead the actual joint readings. This can be seen in Figure 4 (top) at 5.4 s. For the knee joint and for less aggressive trajectories with the hip, the separation is not significant. In addition, note that backlash in the joints is not modeled. The joint encoders or located on the motors rather than the joint axes, which makes it very difficult to measure and model backlash.

## 3.2. Ground Interaction Model

LittleDog is mostly incapable of an aerial phase due to the velocity limits in the joints, so at least one foot is usually in contact with the ground at any time. The ground interaction is complicated, because the robot's foot may roll, slide, stick, bounce, or do some combination of these. A continuous, elastic ground interaction model is used, where the foot of the robot is considered as a ball, and at each point in time the forces acting on the foot are computed. The ground plane is assumed to be compressible, with a stiff non-linear spring damper normal to the ground that pushes the foot out of the terrain. A tangential friction force, based on a non-linear model of Coulomb friction is also assumed. The normal and friction forces are balanced with the force of the shin spring at the bottom of the robot's leg. The rate of change of the shin spring, $\dot{\mathbf{l}}$, is computed by the force balancing and used to update the shin-spring length, $\mathbf{l}$, which is a part of the state space. The resulting normal and friction forces are applied to the five-link model.

Appendix A discusses the details of the foot roll calculation, the normal force model, the friction model, and their

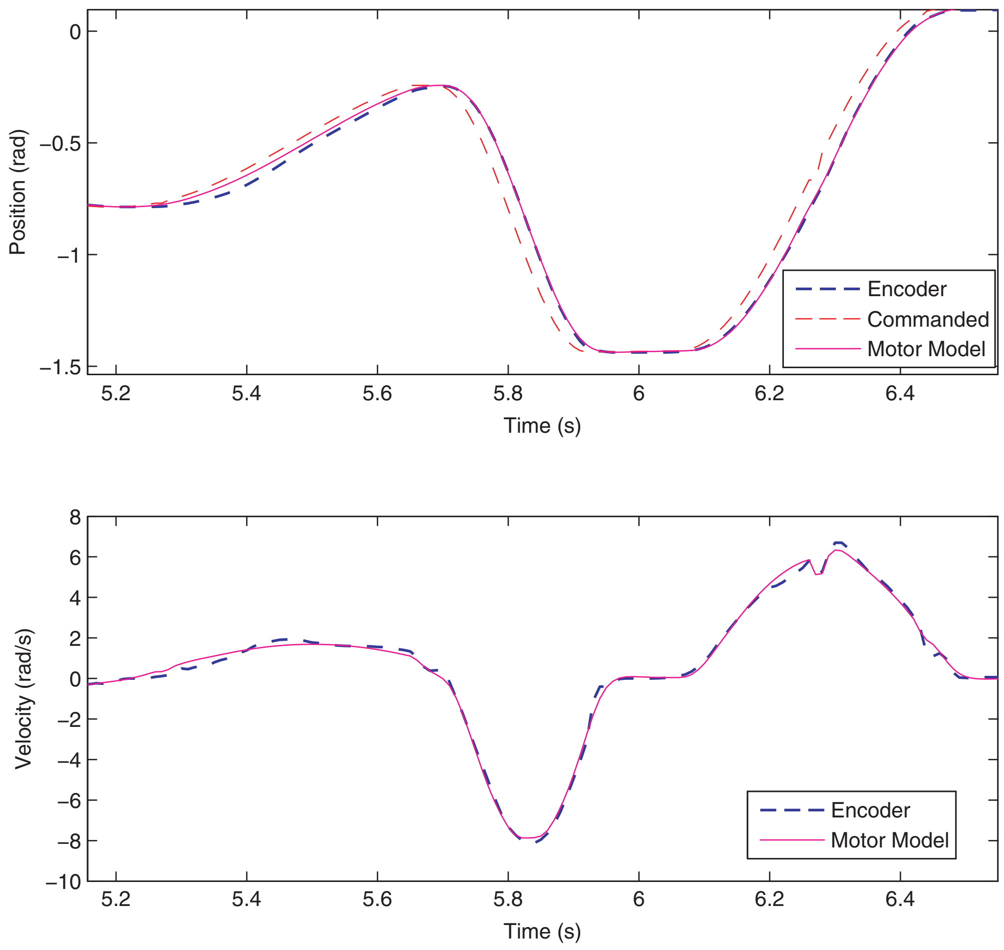

**Fig. 4.** Example of a hip trajectory, demonstrating position command (thin dashed red), motor model prediction (solid magenta), and actual encoder reading (thick dashed blue).

use in computing ground contact forces. The ground contact model is also illustrated in Figure 21 of the Appendix.

## 3.3. *Parameter Estimation*

There are many coupled parameters that determine the behavior of the model. In theory, they could all be fit to a large enough number of robot trajectories, but it would require thoroughly exploring relevant regions of the robot's {STATE-SPACE × ACTION-SPACE}. This is difficult, because LittleDog cannot easily be set to an arbitrary point in state space, and the data we collect only covers a tiny fraction of it. An easier and more robust approach relies on the model structure to separate the fitting into sub-problems and to identify each piece separately. The full dynamical model of the robot consists of the five-link rigid body, the motor model, and the ground force model. A series of experiments and a variety of short bounding trajectories were used to fit the model parameters to actual robot dynamics by minimizing quadratic cost functions over simulation error. Appendix B discusses the approach for the fit and lists the parameters and their values. In total, 34 parameters were measured or fit for the model.

## 3.4. *Model Performance*

Figure 5 shows a comparison of a bounding trajectory in simulation versus 10 runs of the same command executed on the real robot. The simulated trajectory was generated using the planning algorithm described in the next section, which used the developed model. The control input and the starting conditions for all open-loop trajectories in the figure were identical, and these trajectories were not used for fitting the model parameters.

Three of the four plots are of an unactuated coordinate ($x$, $y$, and body pitch), the fourth plot is of the back hip, an actuated joint. The figure emphasizes the difference between directly actuated, position controlled joints compared with unstable and unactuated degrees of freedom. While the motor model tracks the joint positions almost perfectly, even through collisions with the ground, the unactuated coordinates of the open-loop trajectories diverge from each other in less than 2 seconds. Right after completing

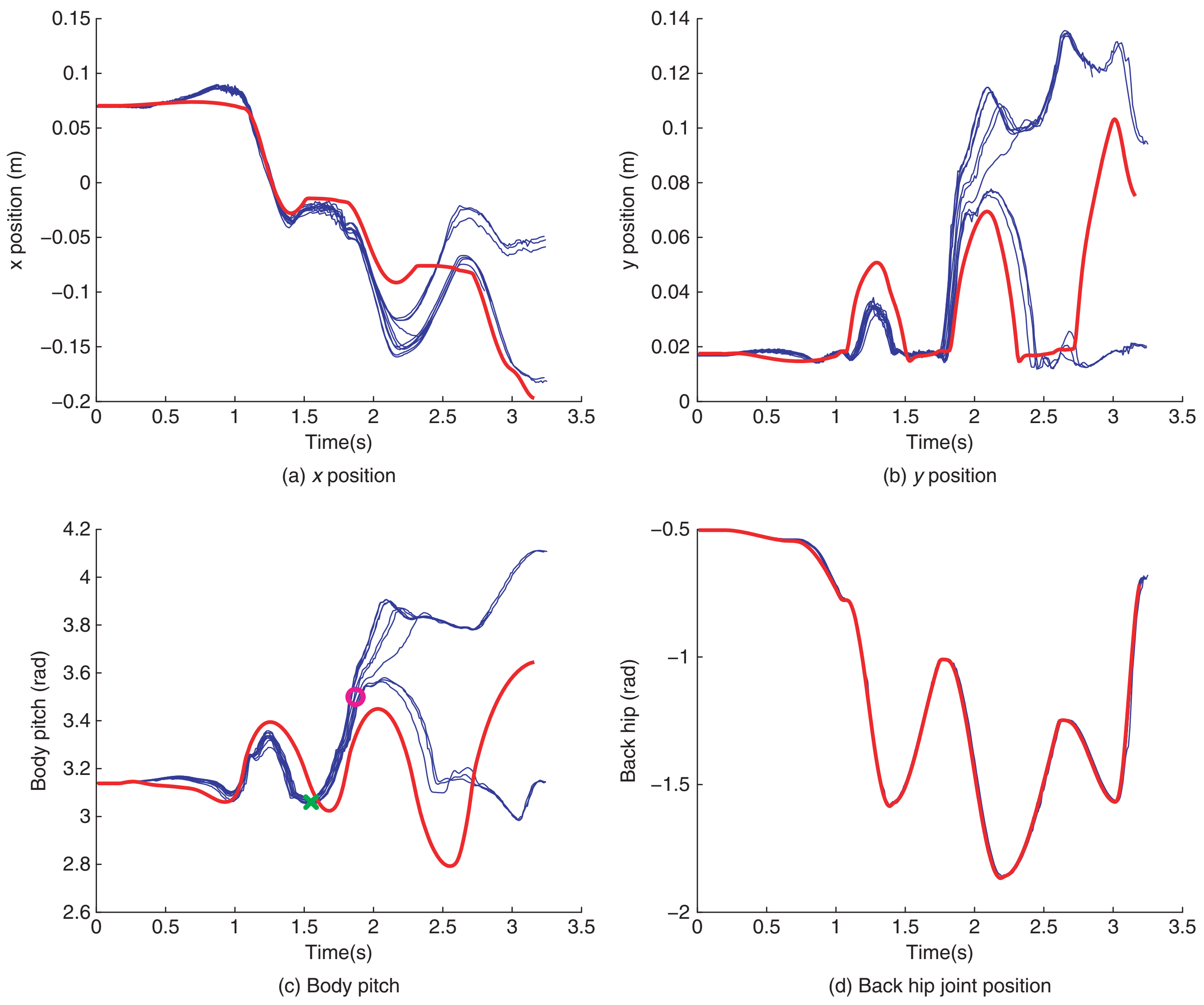

**Fig. 5.** The unactuated coordinates for a bounding motion ($x$, $y$, and body pitch) and a trajectory of one of the joints. The thick red line shows a trajectory generated by planning with RRTs using the simulation model. The thin blue lines are 10 open-loop runs of the same trajectory on a real LittleDog robot. In (c), the “x” shows where trajectories begin to separate, and the “o” show where trajectories finish separating.

the first bounding motion, at about 1.5 s, the trajectories separate as LittleDog is lifting its body on the back feet. At about 1.9 s, in half of the cases the robot falls forward and goes through a second bounding motion, while in the rest of the cases it falls backward and cannot continue to bound. The horizontal position and body pitch coordinates are both highly unstable and unactuated, making it difficult to stabilize them. The control problem is examined in more detail later in this paper.

The most significant unmodeled dynamics in LittleDog include backlash, stiction in the shin spring, and more complex friction dynamics. For example, even though the friction model fits well to steady-state sliding of LittleDog, experiments on the robot show that during a bounding motion there are high-frequency dynamics induced in the legs that reduce the effective ground friction coefficient. Also, the assumption of linearity in the normal force in Coulomb friction does not always hold for LittleDog feet. Modeling these effects is possible, but would involve adding a large number of additional states with non-linear high-frequency dynamics to the model, making it much harder to implement and less practical overall. In addition, the new states would not be directly observable using currently available sensors, so identifying the related parameters and initializing the states for simulation would be difficult.

In general, for a complex unstable dynamical system such as LittleDog, some unmodeled effects will always remain no matter how detailed the model is. Instead of capturing all of the effects, the model approximates the overall behavior of the system, as seen from Figure 5. We believe that this model is sufficiently accurate to generate relevant motion plans in simulation which can be stabilized using feedback on the real robot.

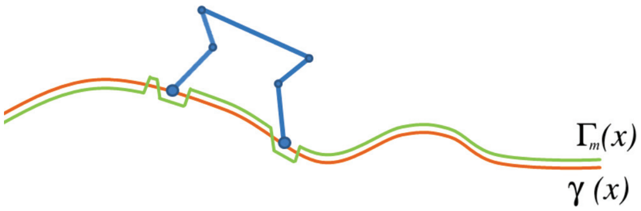

**Fig. 6.** Sketch of a virtual obstacle function, $\Gamma_m(x)$, in relation to the ground, $\gamma(x)$.

## 4. Motion Planning Algorithm

### *4.1. Problem Formulation*

Given the model described in the previous section, we can formulate the problem of finding a feasible trajectory from an initial condition of the robot to a goal region defined by a desired location of the COM. We describe the terrain as a simple height function, $z = \gamma(x)$, parameterized by the horizontal position, $x$. We would like the planned trajectory to avoid disruptive contacts with the rough terrain, however the notion of "collision-free" trajectories must be treated carefully since legged locomotion requires contact with the ground in order to make forward progress.

To address this, we define a virtual obstacle function, $\Gamma(x)$, which is safely below the terrain around candidate foothold regions, and above the ground in regions where we do not allow foot contact (illustrated in Figure 6). In our previous experience with planning walking gaits (Byl et al. 2008; Byl and Tedrake 2009), it was clear that challenging rough terrain could be separated into regions with useful candidate footholds, as opposed to regions where footholds would be more likely to cause a failure. Therefore, we had developed algorithms to pre-process the terrain to identify these candidate foothold regions based on some simple heuristics, and we could potentially use the same algorithms here to construct $\Gamma(x)$. However, in the current work, which makes heavy use of the motion primitive described in the following sections, we found it helpful to construct separate virtual obstacles, $\Gamma_m(x)$, parameterized by the motion primitive, $m$, being performed. Once the virtual obstacles became motion primitive dependent, we had success with simple virtual obstacles as illustrated in Figure 6. The collision function illustrated is defined relative to the current position of the feet. In the case shown in the figure, the virtual function forces the swing leg to lift up and over the terrain, and ensures that the back foot does not slip, which characterizes a successful portion of a bound. As soon as the front feet touch back down to the ground after completing this part of the bound, a new collision function is defined, which takes into account the new footholds, and forces the back feet to make forward progress in the air.

We are now ready to formulate the motion planning problem for LittleDog bounding: find a feasible solution, $\{\mathbf{x}[0], \mathbf{u}[0], \mathbf{x}[1], \mathbf{u}[1], \ldots, \mathbf{x}[N]\}$, which starts in the required initial conditions, satisfies the dynamics of the model, $\mathbf{x}[n+1] = \mathbf{f}(\mathbf{x}[n], \mathbf{u}[n])$, avoids collisions with the virtual obstacles, $\Gamma(\mathbf{x})$, does not violate the bounds on joint positions, velocities, accelerations, and torques, and reaches the goal position.

Given this problem formulation, it is natural to consider a sample-based motion planning algorithms such as RRTs due to their success in high-dimensional robotic planning problems involving complex geometric constraints (LaValle and Branicky 2002). However, these algorithms perform poorly when planning in state space (where the dynamics impose "differential constraints") (Cheng 2005; LaValle 2006), especially in high dimensions. When applied directly to building a forward tree for this problem, they take prohibitive amounts of time and fail to make any substantial progress towards the goal. In the following sections, we describe three modifications to the basic algorithm. First, we describe a parameterized "half-bound" motion primitive which reduces the dimensionality of the problem. Second, we describe the Reachability-Guided RRT, which dynamically changes the sampling distribution and distance metric to address differential constraints and discontinuous motion primitive dynamics. Finally, we describe a mechanism for sampling with a Voronoi bias in the lower-dimensional task space defined by the motion primitive. All three of these approaches were necessary to achieve reasonable run-time performance of the algorithm.

### *4.2. Macro Actions/Motion Primitive*

The choice of action space, e.g. how an action is defined for the RRT implementation, will affect both the RRT search efficiency, as well as completeness guarantees, and, perhaps most importantly, path quality. In the case of planning motions for a five-link planar arm with four actuators, a typical approach may be to consider applying a constant torque (or some other simple action in joint space) that is applied for a short constant time duration, $\Delta T$. One drawback of this method is that the resulting trajectory found by the RRT is likely be jerky. A smoothing/optimization post-processing step may be performed, but this may require significant processing time, and there is no guarantee that the local minima near the original trajectory is sufficiently smooth. Another drawback of using a constant time step with such an action space is that in order to ensure completeness, $\Delta T$ should be relatively small (for LittleDog bounding, 0.1 seconds seems to be appropriate). Empirically, however, the search time increases approximately as $1/\Delta T$, so this is a painful trade-off.

For a stiff PD-controlled robot, such as LittleDog, it makes sense to have the action space correspond directly to position commands. To do this, we generate a joint trajectory by using a smooth function, $\mathcal{G}$, that is parameterized by the initial joint positions and velocities, $[\mathbf{q}(0), \dot{\mathbf{q}}(0)]$, a time for the motion, $\Delta T_m$, and the desired end joint positions and velocities, $[\mathbf{q}_d(\Delta T_m), \dot{\mathbf{q}}_d(\Delta T_m)]$. This action set requires specifying two numbers for each actuated degree

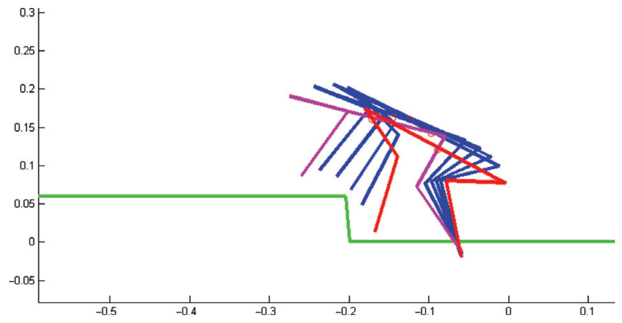

**Fig. 7.** First half of a double bound. The robot is shown moving from right to left. The back leg "opens up", pushing the robot forward (to the left) while generating a moment around the stance foot causing the robot to rear-up. Meanwhile, the front legs tuck-in, and then un-tuck as they get into landing position. Axes are in meters.

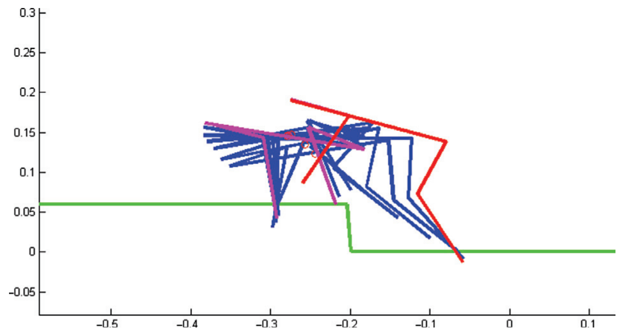

**Fig. 8.** Second half of a double bound. The front (left) foot moves towards the back (right) of the robot, helping to pull the robot forward. The back leg meanwhile swings forward, while tucking and un-tucking to help avoid collision with the terrain.

of freedom: one for the desired end position and one for the desired end velocity. A smooth function generator which obeys the end point constraints, for example a cubic-spline interpolation, produces a trajectory which can be sampled and sent to the PD controller.

If one considers bounding in particular and examines how animals, and even some robots such as the Raibert hoppers, are able to achieve bounding behavior, it becomes apparent that some simplifications can be made to the action space. We can define a motion primitive that uses a much longer $\Delta T$ and, therefore, a shorter search time, while also producing smooth, jerk-free joint trajectories. The insight is based on the observation that a bound consists of two phases: (1) rocking up on the hind legs while moving the front legs forward and (2) rocking up on the front legs, while the hind legs move forward. In the first half-bound primitive, the hind legs begin moving backwards, propelling the body forwards. This forward acceleration of the COM also generates a rotational moment around the pseudo-pin joint at the hind foot–ground interface. In this case, the front legs come off the ground, and they are free to move to a position as desired for landing. In this formulation, the hind legs move directly from the starting pose to the ending pose in a straight line. Because the front feet are light, and not weight bearing, it is useful to "tuck" the feet, while moving them from the start to the end pose, in order to help avoid obstacles. To take advantage of a rotational moment produced by accelerating the stance leg, the back leg begins moving a short time before the front foot leg starts moving. Once both the hind and the front legs have reached their desired landing poses, the remaining trajectory is held constant until the front legs impact the ground. An example of such a trajectory is shown in Figure 7.

A similar approach is utilized to generate motions for the second phase of bounding, with the difference that the hip angles are "contracting" instead of "opening up". As the front leg becomes the stance leg, the front foot begins moving backwards just before impact with the ground. The back leg movement is delayed for a short period to allow the system to start rocking forward on to the front legs. When both legs reach their final positions, the pose is held until the back leg touches down. The back leg tucks in while swinging into landing position to help avoid obstacles. The resulting motions are shown in Figure 8.

Note that for the start and end conditions of the motion primitive, the actuated-joint velocities are zero, a factor which reduces the action space further. Using these motion primitives requires a variable time step, $\Delta T_{\text{bound}}$, because this has a direct influence on accelerations and, therefore, moments around the passive joints. However, for each phase, one only needs to specify four degrees of freedom, corresponding to the pose of the system at the end of the phase. Using this motion primitive, the entire bounding motion is achieved with a simple, jerk-free actuator trajectory. Because these primitives are naturally smooth, post-processing the RRT generated trajectory becomes an optional step.

The motion primitive framework used here is somewhat similar in approach to the work in Frazzoli et al. (2005). In addition to formally defining motion primitives in the context of motion planning, that work proposed planning in the framework of a Maneuver Automaton. The automaton's vertices consist of steady-state trajectories, or trim primitives (in the context of helicopter flight), which are time invariant and with a zero-order hold on control inputs. The edges of the automaton are maneuver primitives, which are constrained so that both the start and the end of the maneuver are compatible with the trim primitives. This is illustrated in the context of LittleDog bounding in Figure 9. Of course, feet collisions in LittleDog act like impulses or hybrid dynamics, which negate the underlying invariance assumptions of a trim primitive in Frazzoli et al. (2005) and Frazzoli et al. (2002), but the idea is still representative.

## *4.3. Reachability-guided RRT Overview*

Sample-based planning methods such as the RRT can be very fast for certain applications. However, such algorithms depend on a distance metric to determine distance from

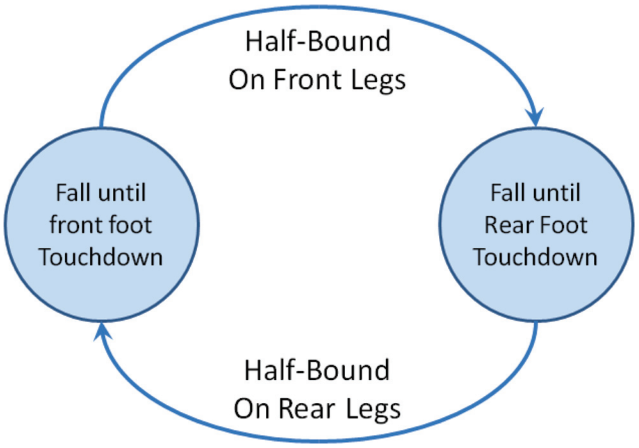

**Fig. 9.** Maneuver automaton for LittleDog bounding.

samples to nodes in the tree. The Euclidean distance metric is easy to implement, and works well for holonomic path planning problems, but it breaks down when constraints imposed by the governing equations of motion restrict the direction that nodes can grow in. To deal with this, we developed a modified version of the RRT algorithm called the Reachability-Guided RRT (RG-RRT) (Shkolnik et al. 2009). The algorithm uses rejection sampling, which was previously shown to be beneficial for growing trees in configuration space through narrow tunnels (Yershova et al. 2005). Reachability guidance extends this idea to planning for dynamic systems in state space. The RG-RRT biases the RRT search toward parts of state space that are locally reachable from the tree. Reachability information is stored in each node in the tree; the algorithm works by sampling and then measuring the Euclidean (or scaled-Euclidean) distance to each node in the tree, as well as to each reachable set of the tree. If a sample is closer to the tree itself, rather than to the reachable set of the tree, then there are no actions which are able to extend that node in the direction of the sample. This sample–node pair is therefore not useful for exploration, and is discarded. On the other hand, if the sample is closer to a reachable region, the parent node in the tree, which generated the closest reachable region, is grown towards the sample. Thus, by keeping track of locally reachable parts of the tree and using a simple Euclidean-type distance metric, the algorithm is able to overcome many of the inherent difficulties when implementing an RRT for dynamic systems.

In this work, we show that the RG-RRT algorithm can be useful for motion planning in a system with motion primitives. Conceptually, a motion primitive in a dynamic system is one which occurs over a fairly substantial duration of time, e.g. an action which moves the system from one state to another state that is relatively distant from the current state. Therefore, the states produced by taking macro actions or motion primitives may be discontinuous or even discrete in relation to a generating state, which would invalidate the use of a proper Euclidean distance metric that assumes a smooth continuous space of actions. The idea of Reachability can be an especially powerful notion for sample-based planning in the context of macro actions, as the reachable region does not have to be local to its parent node. A continuous motion primitive action space can be approximated by sampling in the action space, resulting in a discrete set of reachable nodes, which might be far in state space from their parent node. This idea suggests that the RG-RRT also naturally extends to models with hybrid dynamics, simply by sampling actions and integrating through the dynamics when generating the reachable set.

**Definition 1.** *For a state $x_0 \in \mathcal{X}$, we define its* reachable set, *$\mathcal{R}_{\Delta t}(x_0)$, to be the set of all points that can be achieved from $x_0$ in bounded time, $\Delta t \in [\Delta T_{\text{low}}, \Delta T_{\text{high}}]$, according to the state equations* (1) *and the set of available control inputs, $\mathcal{U}$.*

**Algorithm 1** $\mathcal{T} \leftarrow$ BUILD-RG-RRT($x_{\text{init}}$)

```
1:  $\mathcal{T} \leftarrow$ INITIALIZETREE($x_{\text{init}}$);
2:  $\mathcal{R} \leftarrow$ APPROXR( [ ], $\mathcal{T}, x_{\text{init}}$);
3:  for $k = 1$ to $K$ do
4:    RejectSample $\leftarrow$ true;
5:    while RejectSample do
6:      $x_{\text{rand}} \leftarrow$ RANDOMSTATE( );
7:      $([\,], dist_{\mathcal{T}}) \leftarrow$ NEARESTSTATE($x_{\text{rand}}, \mathcal{T}$);
8:      $(r_{\text{near}}, dist_{\mathcal{R}}) \leftarrow$ NEARESTSTATE($x_{\text{rand}}, \mathcal{R}$);
9:      if $dist_{\mathcal{R}} < dist_{\mathcal{T}}$ then
10:       RejectSample $\leftarrow$ false;
11:       $x_{\text{near}} \leftarrow$ ParentNode($r_{\text{near}}, \mathcal{R}, \mathcal{T}$);
12:     end if
13:   end while
14:   $u \leftarrow$ SOLVECONTROL($x_{\text{near}}, x_{\text{rand}}$);
15:   $[x_{\text{new}}, isFeasible] \leftarrow$ NEWSTATE($x_{\text{near}}, u$);
16:   if isFeasible then
17:     $\mathcal{T} \leftarrow$ INSERTNODE($x_{\text{new}}, \mathcal{T}$);
18:     $\mathcal{R} \leftarrow$ APPROXR($\mathcal{R}, \mathcal{T}, x_{\text{new}}$);
19:     if ReachedGoal($x_{\text{new}}$) then
20:       return $\mathcal{T}$
21:     end if
22:   end if
23: end for
24: return [ ]
```

The structure of the RG-RRT algorithm is outlined in Algorithm 1. Given an initial point in state space, $x_{\text{init}}$, the first step is to initialize the tree with this state as the root node. Each time a node is added to the tree, the APPROXR( ) function solves for $\mathcal{R}_{\Delta t}(x_{\text{new}})$, an approximation of the set of reachable points in the state space that are consistent with the differential constraints (1). The approximated reachable set for the whole tree is stored in a tree-like structure, $\mathcal{R}_{\Delta t}(\mathcal{T})$, or simply $\mathcal{R}$ for shorthand notation, which contains reachable set information as well as pointers to the

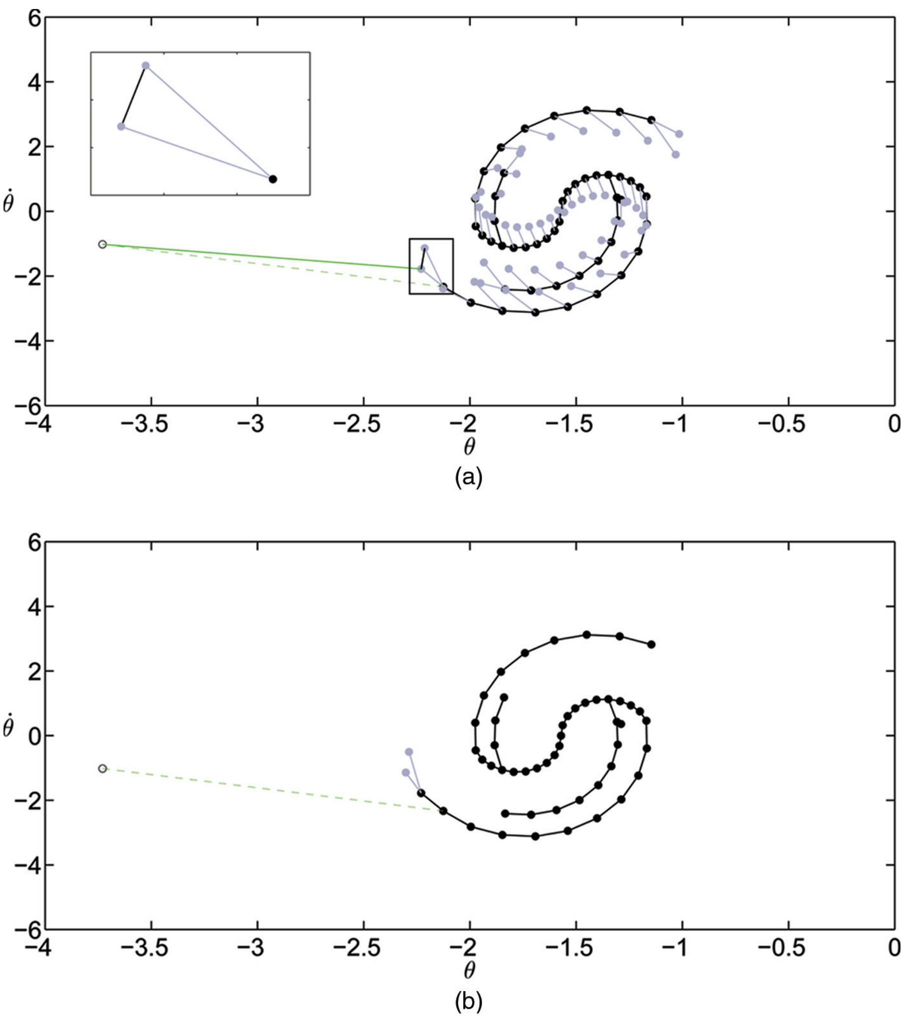

**Fig. 10.** Two consecutive steps in a single iteration of the RG-RRT, as demonstrated for the underactuated pendulum. In (a), a random sample is drawn from the state space and paired with the nearest node in the tree and the closest point in its reachable set according to the Euclidean distance metric. Shown in (b), the algorithm then expands the node towards the point in the reachable set and adds that point to the tree. Only the reachable region of the newly added point is shown here for clarity.

parent nodes for each node in the corresponding tree, $\mathcal{T}$. For many systems of interest, the approximate bounds of the reachable set, $\mathcal{R}_{\text{hull}}$, can be generated by sampling over the limits in the action space, and then integrating the corresponding dynamics forward. For these systems, assuming a relatively short action time step, $\Delta t$, integrating actions between the limits results in states that are within, or reasonably close to being within, the convex hull of $\mathcal{R}_{\text{hull}}$. In such cases it is sufficient to consider only the bounds of the action set to generate the reachability approximation. When the dimension of the action space becomes large, it may become more efficient to approximate the reachable set with a simple geometric function, such as an ellipsoid. We found that the reachable set approximation does not need to be complete, and even a crude approximation of the set can vastly improve planner performance in systems with dynamics. Another benefit of approximating the reachable set by sampling actions is that the resulting points can be efficiently tested for collisions before they are added to the reachable set approximation. This reduces the likelihood of trajectories leaving free space as part of the exploration phase.

After a node and its corresponding reachable set is added to the tree, a random state-space sample, $x_{\text{rand}}$, is drawn. The NEARESTSTATE($x_{\text{rand}}, [\mathcal{T}, \mathcal{R}]$) function compares the distance from the random sample to either the nodes of the tree, $\mathcal{T}$, or the reachable set of the tree, $\mathcal{R}$. The function returns the closest node as well as the distance from the node to the sample. Samples that are closer to the tree, $\mathcal{T}$, rather than the reachable set, $\mathcal{R}$, are rejected. Sampling continues until a sample is closer to $\mathcal{R}$, at which point, the *ParentNode* function returns the node in $\mathcal{T}$, which is the parent of the closest reachable point in $\mathcal{R}$.

Consider planning on a simple torque-limited pendulum, which has interesting dynamics, and can be easily visualized, because the state space is two dimensional. A single

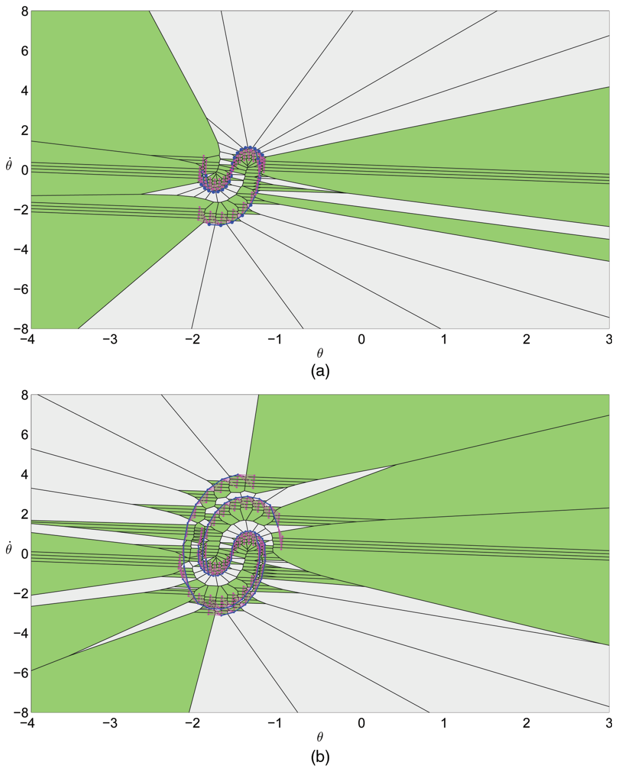

**Fig. 11.** RG-RRT Voronoi diagrams for a pendulum (best viewed in color online). The diagram in (a) corresponds to the RG-RRT after 30 nodes have been added while that in (b) corresponds to the tree with 60 nodes. Magenta dots are discretely sampled reachable points affiliated with each tree. Green regions are areas where samples are "allowed", and correspond to Voronoi areas associated with the reachable set. Samples that fall in gray areas are discarded.

expansion step of the RG-RRT is shown in Figure 10. The tree nodes are shown in black, with the reachable approximation shown in gray. Figure 11 shows the associated Voronoi regions of the combined set of $\{\mathcal{R}, \mathcal{T}\}$ for an RG-RRT growing on a torque-limited pendulum. The rejected sampling region is shown in gray, while the allowed regions are shown in green. Note that the sampling domain in the RG-RRT is dynamic, and adapts to the tree as the tree expands, producing a Voronoi bias that is customized by the system dynamics defined by the tree. Also note that although the Euclidean distance is used, we are warping this distance metric by computing the distance to the reachable set rather than to the tree. This warped metric produces a Voronoi bias to "pull" the tree in directions in which it is capable of growing.

## 4.4. Approximating the Reachable Set

We have shown that Reachability Guidance can improve RRT planning time in underactuated systems such as the torque-limited pendulum and Acrobot (Shkolnik et al. 2009). In those systems, the reachable set was approximated by discretizing the action space. However, to achieve LittleDog bounding, even a reduced four-dimensional action space becomes too large to discretize efficiently. For example, using only three actions per dimension (and assuming a constant motion primitive time step), there would be 81 actions to apply and simulate in order to approximate the reachable set for each node.

Instead of discretizing in such a high-dimensional action space, the reachable set can be approximated by understanding the failure modes of bounding. Failures may occur primarily in one of three ways: (1) the robot has too much energy and falls over; (2) the robot has too little energy, so the stance foot never leaves the ground, violating our assumption that one foot always leaves the ground in a bounding gait; or (3) a terrain collision occurs. In the case when the system has a lot of kinetic energy, the best thing to do is to straighten all of the limbs, which raises the COM and converts the kinetic energy into potential energy. At the other extreme, if the robot does not have much kinetic energy, an action that lowers the COM while accelerating the limbs inwards tends to produce a rotational moment, if it is possible to do so. Thus, two extreme motions can be generated, for each phase of bounding, which prevent the two common failure modes. The reachable set is then generated by interpolating joint positions between the two extremes. This one-dimensional discretization is usually rich enough to capture the energy related failure modes and generate bounds which strike the right balance to continue bounding further.

The reachable set helps the RG-RRT to plan around Regions of Inevitable Collisions (RIC) (LaValle and Kuffner 2001; Fraichard and Asama 2003; Fraichard 2007; Chan et al. 2008). Nodes which have empty reachable sets, which may occur because the node has too much or too little energy, in the case of LittleDog bounding, are in $\mathcal{X}_{\text{RIC}}$, even if the node itself is collision free ($\mathcal{X}_{\text{free}}$). The RG-RRT takes advantage of this, because when a node has an empty reachable set, the node serves as a "blocking" function for sampling. Owing to the rejection sampling of the RG-RRT, such a node cannot be expanded upon, and any samples that map to this node will be discarded, encouraging sampling in other areas that can be expanded.

## 4.5. Sampling in Task Space

In the RRT algorithm, sampling is typically performed uniformly over the configuration space. An action is chosen which is expected to make progress towards the sample. The sampling creates a Voronoi bias for fast exploration by frequently selecting nodes of the tree near unexplored regions, while occasionally refining within explored regions. We have previously shown that the Voronoi bias can be implemented in the configuration (or state) space or alternatively the bias can be in a lower-dimensional task space (Shkolnik and Tedrake 2009). This can be achieved simply by sampling in task space, and finding the node in the configuration-space tree, the projection of which is closest to the task-space sample. Reducing the dimensionality of the search with a task-space Voronoi bias can significantly improve search efficiency and, if done carefully, does not have an impact on the completeness of the algorithm.

As described in the previous section, our action space involves a half-bound (half a period of a complete bound). At the start and end of an action (e.g. the state at any given node on the tree), the robot is approximately touching the ground with both feet, and joint velocities are approximately zero. Samples are therefore similarly constrained. In addition, samples are chosen such that they are not in collision, and respect joint bounds, with some minimum and maximum stance width. The region in actuated joint space can be mapped to a region in Cartesian space for the back and front foot, corresponding to a four-dimensional manifold. A sample is drawn by first choosing a horizontal coordinate, $x$, of the robot's back foot, and then selecting four joint angles from the manifold, while checking to ensure that collision constraints are validated.

Given a sample described above, the $y$-coordinate of the robot foot is set to the ground position at $x$, and the passive joint is computed using the constraint that both feet are on the ground. Thus, sampling the five-dimensional space maps to a point $\mathbf{q}_s$ in the seven-dimensional configuration space of the robot. The five-dimensional sampling space, which neglects velocities, is significantly smaller than the complete 16-dimensional state space, and produces a task-space Voronoi Bias. When a sample is drawn, the closest node is found by minimizing the Euclidean distance from the sample to the tree, as well as to the reachable region of the tree. The sampling is repeated until a sample closest to the reachable region is found. An action, $\mathbf{u}$, is then created by selecting the four actuated joint angles from the sample, $\mathbf{q}_s$. An action time interval $\Delta T_{\text{bound}}$ is chosen by uniformly sampling from $T \in [0.3, 0.7]$ seconds.

## 4.6. Simulation Results

In this section, we have presented three modifications to the standard implementation of the RRT to plan bounding motions with LittleDog: (1) a simple motion primitive; (2) reachability guidance; and (3) task-space biasing. Each of these components could be implemented separately, but they worked particularly well when combined. To show this, we qualitatively compare results by applying various combinations of these modifications.

First, a standard RRT was implemented, without any modifications, with the task of finding bounding motions on flat terrain. The state space was sampled uniformly in the regions near states that have already been explored. We experimented with several types of action spaces, including a zero-order hold on accelerations in joint space, or a zero-order hold on velocities in joint space. Reasonable results were found by specifying a change in joint positions ($\Delta q_a$), and using a smooth function generator to create position splines with zero start and end velocities. This approach worked best using time intervals of 0.1–0.15 seconds, but the resulting RRT was not able to find a complete bounding motion plan in a reasonable amount of time (hours). By implementing reachability-guidance and task-space sampling, the planner was able to find a double-bound trajectory on flat terrain, after 2–3 minutes of search on average. Some of these plans were successfully executed on flat terrain by the actual robot. In addition to being relatively slow to plan motions, the trajectories returned by the RRT had high-frequency components which were difficult for the robot to execute without saturating motors. The long planning time and sub-optimal gaits were not ideal for this task. Plans might be smoothed and locally optimized, but this is a time-consuming process given the complexity of the dynamics.

When we utilized the half-bound motion primitive, the trajectories produced were smooth by the way the primitive is defined, with one hip movement and one knee movement per each half-bound motion. Furthermore, the half bound has a relatively long duration, typically 0.5 seconds in length, which shortens the search time. The complete planner described in this section is able to plan a double-bound on flat terrain in a few seconds. A continuous bound over 2 m is typically found in less than a minute. The complete planner was also able to plan over intermittent terrain, where foot holds were not allowed in given regions. In addition, the planner was successful in handling a series of 7-cm steps, and was also successful in planning bounds to get up on top of a terrain with artificial logs with a maximum height difference of 8 cm. The simulated log terrain corresponds to a laser scan of real terrain board used in the Learning Locomotion program to test LittleDog performance. An example of a bounding trajectory is shown in Figures 12 and 13, and a video of this trajectory is included in Extension 1. The bottom link in the robot leg is 10 cm, and the top link is 7.5 cm; given that the bottom of the body is 3 cm below the hip, variations of 7 cm in terrain height represents approximately 50% of maximum leg travel of the robot.

Planning was also attempted using the motion primitive, but without reachability guidance. To do so, samples were drawn in each iteration of the RRT algorithm, and the nearest-neighbor function returned the closest node in the tree. Because the closest node in the tree and the sample often looked similar, it did not make sense to try to expand towards the sample. Instead, once a sample was chosen, a motion primitive action was chosen at random. Using random actions is an accepted approach for RRTs, and the sampling itself produces a Voronoi bias to encourage the tree to expand into unexplored regions of space, on average. In this case, however, the motion primitive-based RRT was never able to find feasible plans over rough terrain without reachability guidance, even when given over 12 hours to do so.

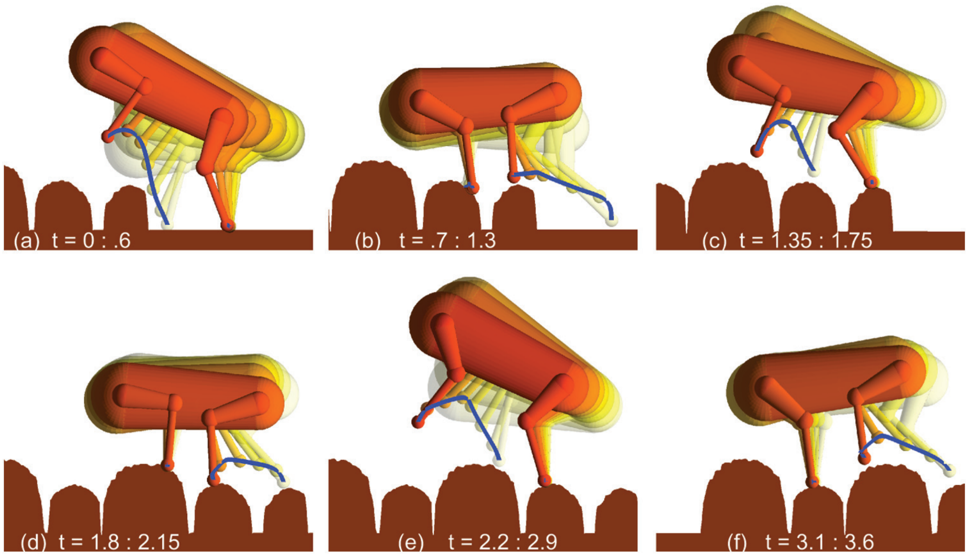

**Fig. 12.** Bounding trajectory over logs (Part 1).

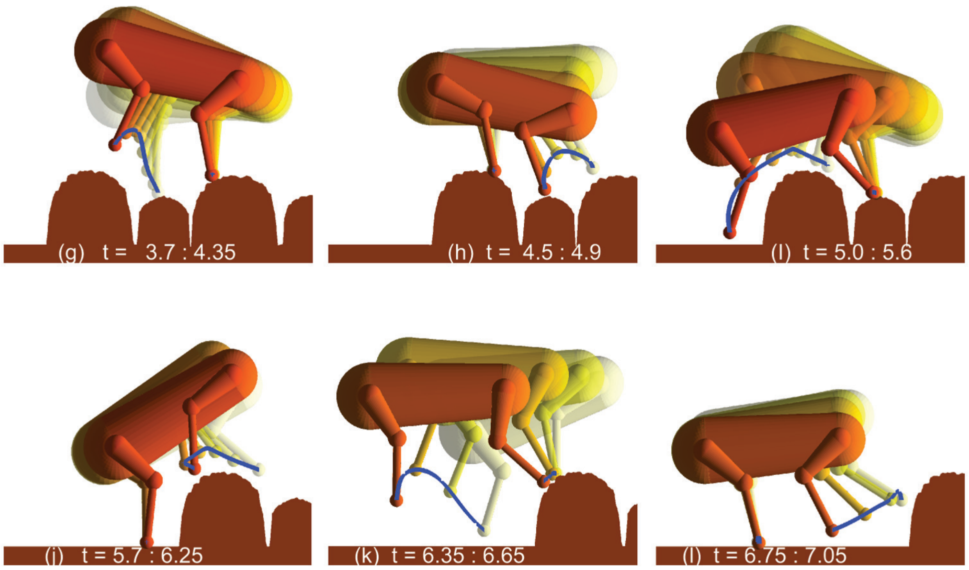

**Fig. 13.** Bounding trajectory over logs (Part 2).

The combination of motion primitives, task-space biasing and reachability guidance, however, is able to plan trajectories over the log terrain repeatedly, within 10–15 minutes of planning time required on average. Of course, on simpler terrain, the planning time is much faster.

## *4.7. Experimental Results*

In experiments with the real robot, open-loop execution of the motion plan found by the RRT quickly diverged with time from the model predictions, even on flat terrain. Trajectories are unstable in the sense that given the same initial conditions on the robot and a tape of position commands to execute, the robot trajectories often diverge, sometimes catastrophically. To demonstrate the feasibility of using the motion primitives described in this paper, these motion primitives were used to achieve a short bound sequence over the log terrain. Unlike in the model, the actual logs are not planar. To address this, tracks were laid on the logs corresponding to the stance width of the robot along the terrain. The planner was constrained to only allow foot holds where the points on adjacent tracks were of the same height. With some tuning, the motion primitives produced bounding over the logs on the real robot, shown in Figure 14. This trajectory was successful approximately 20% of the time, even though care was taken to have the same initial position for each trial.

Feedback stabilization is required to enable execution of long bounding trajectories. Feedback can compensate for model errors and the inherent instability of the system itself.

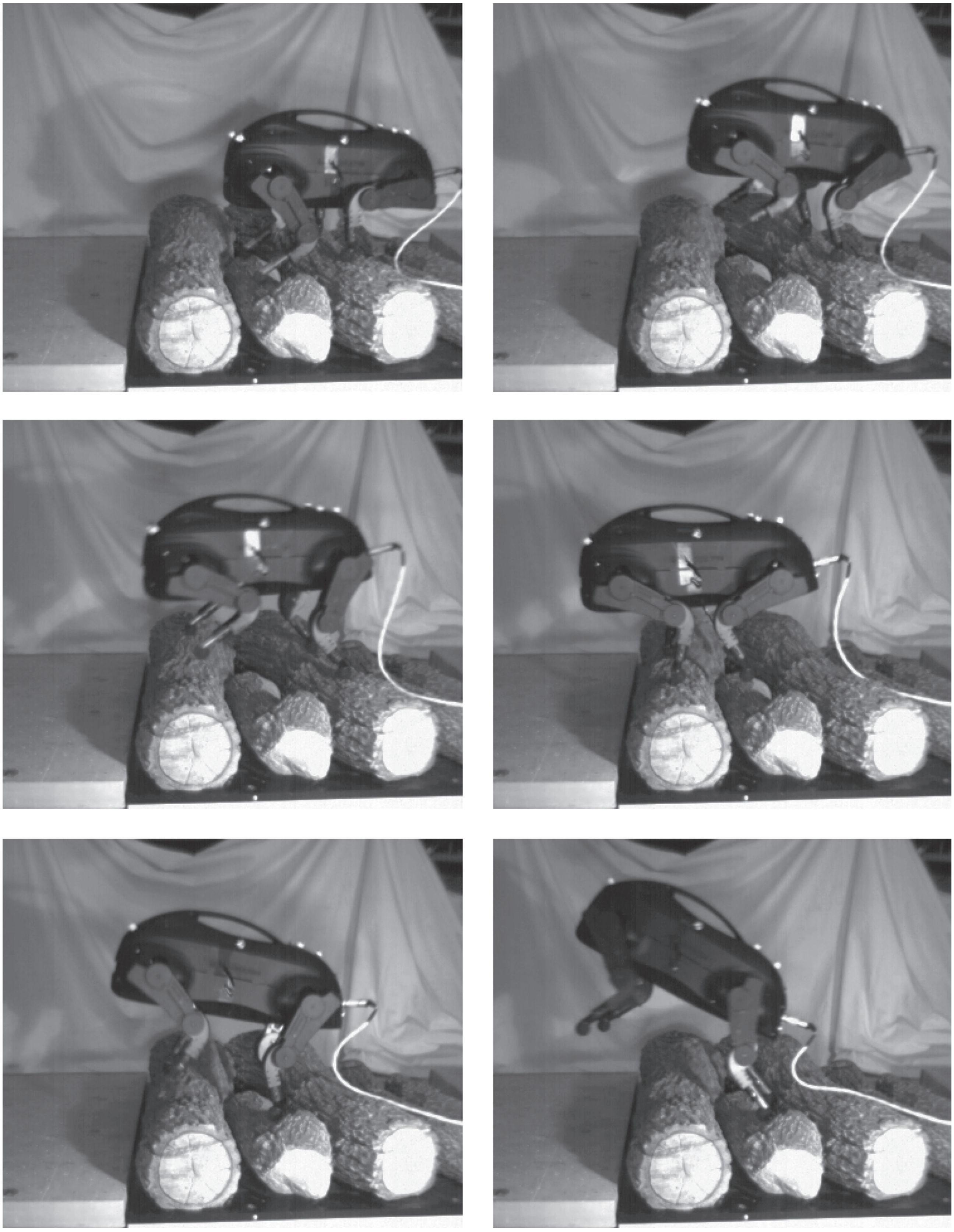
**Fig. 14.** Bounding over logs with LittleDog.

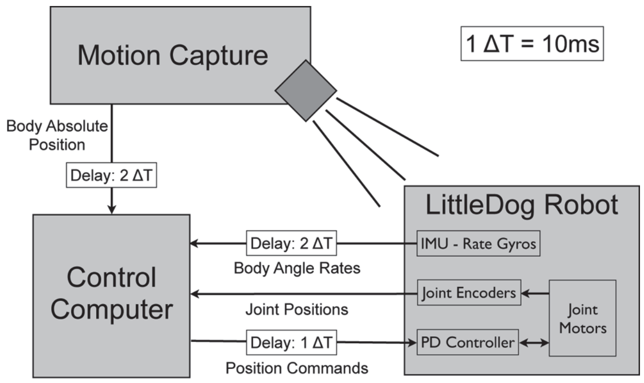

**Fig. 15.** The sensing and control environment for LittleDog.

Trajectory optimization may also help to improve performance. Care was taken to ensure that the simulation is smooth and continuous, so gradients can be used to help in these tasks. The next section describes a feedback control strategy, and discusses the feasibility for this strategy to stabilize bounding gaits on rough terrain.

## 5. Feedback Stabilization

Figure 15 shows the sensing and control environment for LittleDog. For most feedback control methods, one selects a state space suitable for the task at hand, and uses an observer to construct an estimate of the current state using available measurements. An accurate observer requires a good dynamical model of the robot and knowledge of the characteristics (e.g. delays and noise) of each measurement.

Having estimated the state, the controller must select an action to take, which stabilizes the bounding motion. This is a non-trivial task because of the highly non-linear nature of the dynamics and the limited control authority available for the unactuated coordinate. Several recent efforts in control of underactuated systems have found success in taking a three-stage approach (Westervelt et al. 2003; Song and Zefran 2006; Westervelt et al. 2007; Shiriaev et al. 2008; Manchester et al. 2009):

1. From encoder, inertial measurement unit (IMU), and motion-capture measurements, estimate the current state of the system in terms of generalized coordinates $\hat{\mathbf{x}}(t) = (\hat{q}(t), \hat{\dot{q}}(t))$.
2. Based on the current state estimate $\hat{\mathbf{x}}(t)$, find the location on the planned trajectory which is "closest" in some reasonable sense. That is, perform a *projection* to a point in the planned motion $\mathbf{x}^\star(\tau)$ where $\tau$ is computed as some function of $\hat{\mathbf{x}}(t)$.
3. From the deviation $\hat{\mathbf{x}}(t) - \mathbf{x}^\star(\tau)$, and some precomputed local control strategy, compute control actions which bring the system closer to the desired trajectory.

For example, in the method of *virtual constraints* for biped control (Westervelt et al. 2003, 2007), the planned trajectory is parametrized in terms of a phase variable which is often the unactuated coordinate (e.g. the ankle angle), or some function of all coordinates (e.g. angle from foot contact point to hip), which is monotonic over each step and can thus be considered a reparametrization of time. The projection (step 2) is done by computing the current value of the phase coordinate and finding the unique point on the planned trajectory with the same value. The positions of all actuated joints are then synchronized to this joint via high-gain feedback. This method has been successful in biped control (Chevallereau et al. 2003) but poses challenges for quadruped bounding due to the lack of a configuration variable that evolves monotonically over each half-bound and can be used as a phase variable.

An alternative approach is the *transverse linearization*, in which the projection can be a more general function. Transverse linearizations are a classical tool by which to study stability of periodic systems (Hahn 1967; Hale 1980; Hauser and Chung 1994) and more recently have been used for constructive controller design (Song and Zefran 2006; Shiriaev et al. 2008; Manchester et al. 2009; Manchester 2010). The $n$-dimensional dynamics around the target trajectory are decomposed into two parts: (1) a scalar "phase variable" $\tau$ which represents the position along the trajectory and can be considered a reparametrization of time; and (2) an $(n-1)$-dimensional vector $\mathbf{x}_\perp$ representing dynamics transverse to the target trajectory.

In some region around the target orbit, the continuous-time dynamics in the new coordinate system are well defined and have the form

$$\dot{\mathbf{x}}_\perp = \mathbf{A}(\tau)\mathbf{x}_\perp + \mathbf{B}(\tau)\delta\mathbf{u} + \mathbf{h}(\mathbf{x}_\perp, \tau), \tag{3}$$
$$\dot{\tau} = 1 + \mathbf{g}(\mathbf{x}_\perp, \tau). \tag{4}$$

Here $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}^\star(\tau)$, $\mathbf{h}(\cdot)$ contains terms second order and higher in $\mathbf{x}_\perp$, and $\mathbf{g}(\cdot)$ contains terms first order and higher in $\mathbf{x}_\perp$ and $\tau$.

The first-order approximation to the transverse component is known as the *transverse linearization*:

$$\dot{\mathbf{x}}_\perp = \mathbf{A}(\tau)\mathbf{x}_\perp + \mathbf{B}(\tau)\delta\mathbf{u}. \tag{5}$$

Exponential stabilization of the transverse linearization is equivalent to orbital exponential stabilization of the original non-linear system to the target motion. A construction based on a Lagrangian model structure (Shiriaev et al. 2008) has been used to stabilize non-periodic motions of an underactuated biped over rough terrain, and validated in experiments (Manchester et al. 2009). Our model of LittleDog includes highly non-linear compliance and ground-contact interactions, and does not fit in the Lagrangian framework. To derive the controller we used a more general construction of the transverse linearization which will be presented in more detail elsewhere (Manchester 2010).

To stabilize LittleDog, we discretized the transverse linearization (5) to a zero-order-hold equivalent, and computed controller gains using a finite-time LQR optimal control on the transverse linearization, i.e. the controller minimizing the following cost function:

$$J = \sum_{k=0}^{T} \left[ \mathbf{x}_\perp(k)' \, \mathbf{Q}(k) \, \mathbf{x}_\perp(k) + \delta\mathbf{u}(k)' \, \mathbf{R}(k) \, \delta\mathbf{u}(k) \right] + \mathbf{x}(T)' \, \mathbf{Q}_F \mathbf{x}(T)$$

where $\mathbf{Q}(\cdot), \mathbf{R}(\cdot)$, and $\mathbf{Q}_F$ are weighting matrices. This gives a sequence of optimal controller gains:

$$\delta\mathbf{u}(k) = -\mathbf{K}(k) \, \mathbf{x}_\perp(k)$$

which are computed via the standard Riccati difference equation.

From the combination of a projection $P$ from $\mathbf{x}$ to $(\tau, \mathbf{x}_\perp)$ and the LQR controller, one can compute a controller for the full non-linear system:

$$(\tau, \mathbf{x}_\perp) = P(\mathbf{x}), \tag{6}$$

$$\mathbf{u} = \mathbf{u}^\star(\tau) - \mathbf{K}(\tau) \, \mathbf{x}_\perp, \tag{7}$$

where $\mathbf{K}(\tau)$ is an interpolation of the discrete-time control gain sequence coming from the LQR. We refer to this as the "transverse-LQR" controller.

Note that the above strategy can be applied with a receding-horizon and particular final-time conditions on the optimization in order to ensure stability (Manchester et al. 2009). We found with LittleDog, however, that relaxing these conditions somewhat by performing a finite-horizon optimization over each half-bound, with $\mathbf{Q}_F = 0$, was sufficient, even though it is not theoretically guaranteed to give a stabilizing controller. In this framework, the transverse-LQR controller is applied until just before impact is expected. During the impact, the tape is executed open loop, and the next half-bound controller takes over shortly after impact.

### 5.1. Selection of Projection and Optimization Weights

Using the transverse-LQR controller we were able to achieve stable bounding in simulations with several types of perturbations. The success of the method is heavily dependent on a careful choice of both the projection $P(x)$ and the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$.

For the projection, LittleDog was represented as a floating five-link chain but in a new coordinate system: the first coordinate was derived by considering the vector from the point of contact of the stance foot and the COM of the robot. The angle $\theta$ of this line with respect to the horizontal was the first generalized coordinate. The remaining coordinates were the angles of the four actuated joints, and the $(x, y)$ position of the non-stance (swing) foot. In this coordinate system, the projection on to the target trajectory was taken to be the closest point under a weighted Euclidean distance, with heaviest weighting on $\theta$, somewhat lighter weighting on $\dot{\theta}$, and much lighter weighting on the remaining coordinates.

Projecting in this way resulted in significantly improved robustness compared with projection in the original coordinates under a standard Euclidean metric. One can explain this intuitively in terms of controllability: the distance to a point on the target trajectory should be heavily weighted in terms of the states which are most difficult to control. The angles of the actuated joints and the position of the swing foot can be directly controlled at high speed. Therefore, deviations from the nominal trajectories of these coordinates are easy to correct and these coordinates are weighted lightly. In contrast, the angle of the COM and its velocity are not directly controlled and must be influenced indirectly via the actuated links, which is more difficult. Therefore, deviations in these coordinates have a much heavier weighting.

For choosing the optimization weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ we have a similar situation: the dynamics of LittleDog are dominated on $\theta$ and $\dot{\theta}$. Roughly speaking, these states represent its "inverted pendulum" state, which is a common reduced model for walking and running robots. Although we derive a controller in the full state space, the optimization is heavily weighted towards regulating $\theta$ and $\dot{\theta}$.

### 5.2. Simulation Results

We have simulated the transverse-linearization controller with a number of RRT-generated bounding trajectories. For some trajectories over flat terrain in simulation, we were able to stabilize these trajectories even when adding a combination of: (1) slight Gaussian noise to the state estimate ($\sigma = 0.01$ for angles, $\sigma = 0.08$ for velocities), (2) significant velocity perturbations up to 1 rad s$^{-1}$ after each ground impact, (3) model parameter estimation error of up to 1 cm in the main body COM position, and (4) delays of up to 0.04 seconds. In this section we will show some results of stabilization on two example terrains: bounding up stairs and bounding over logs.

Since the impact map is the most sensitive and difficult-to-model phase of the dynamics, we can demonstrate the effectiveness of the controller by adding normally distributed random angular velocity perturbation to the passive (stance–ankle) joint after each impact. Figure 16 shows example trajectories of LittleDog bounding up stairs (perturbations with standard deviation of 0.2 rad s$^{-1}$), and Figure 17 shows it bounding over logs (perturbations with standard deviation of 0.1 rad s$^{-1}$).

In each figure, the trajectory of the COM is plotted for three cases: (1) the nominal (unperturbed) motion, as computed by the RRT planner, (2) running the nominal control inputs open-loop with passive-joint velocity perturbations, and (3) the transverse-LQR stabilized robot with the same perturbations. One can see that for both terrains, after the first perturbation, the open-loop robot deviates wildly and

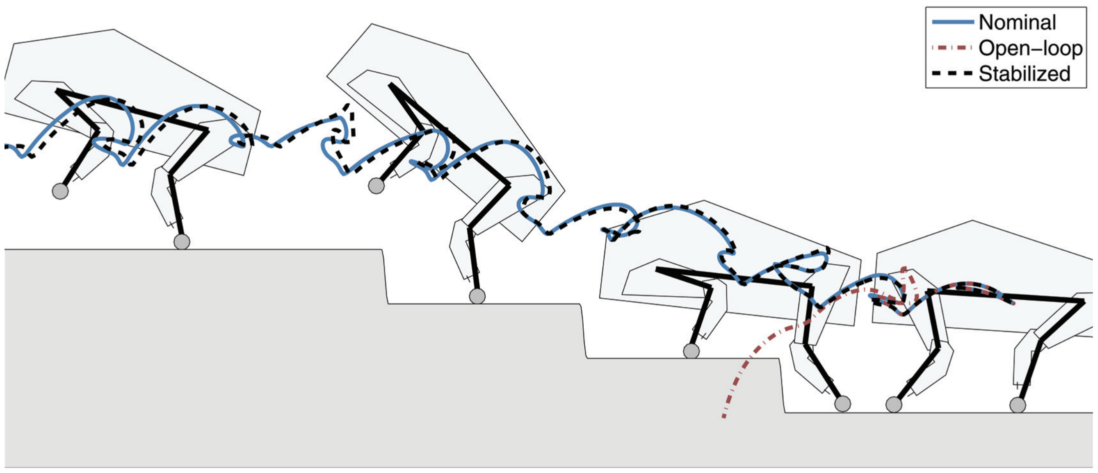

**Fig. 16.** Illustration of bounding up steps with center-of-mass trajectories indicated for nominal, open loop with perturbations, and stabilized with perturbations.

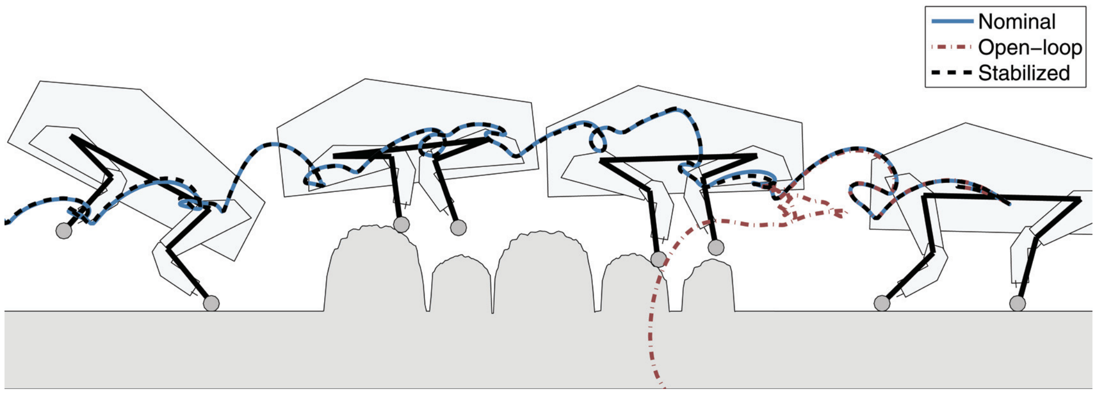

**Fig. 17.** Illustration of bounding over logs with center-of-mass trajectories indicated for nominal, open loop with perturbations, and stabilized with perturbations.

falls over. This shows the inherent instability of the motion. In contrast, the stabilized version is able to remain close to the nominal trajectory despite the perturbations. Videos of these trajectories can be found in Multimedia Extension 1.

We analyze this behavior in more detail in Figure 18. Here we depict the phase portrait of the COM angle $\theta$ and its derivative $\dot{\theta}$ during a single bound for the same three cases. This coordinate is not directly actuated, and can only be controlled indirectly via the actuated joints. Note that the nominal trajectory comes quite close to the state $\theta = \pi/2, \dot{\theta} = 0$. This state corresponds to the robot being balanced upright like an inverted pendulum, and is an unstable equilibrium (cf. the separation of trajectories in Figure 5). One can see that when running open-loop, a small perturbation in velocity pushes the robot to the wrong side of this equilibrium, and the robot falls over. In contrast, the transverse-LQR stabilized robot moves back towards the nominal trajectory.

## *5.3. Experimental Results*

We are currently working to implement the transverse-LQR system on the real robot. At present, the main difficulty stems from the measurement system on the experimental platform, seen in Figure 15. In order to implement our controller, we must have knowledge of the state of the robot, and in experiments this is derived from a combination of an on-board IMU, motor encoders, and the motion-capture system.

The motion-capture system has quite large delays, of the order of 0.02–0.03 s, and the IMU is noisy. Furthermore, the joint encoders measure position of the motors, which can be substantially different to the true joint positions due to backlash. When we implement our controller on the real robot, we typically see large, destabilizing oscillations. We can reproduce similar oscillations in our simulation by including significant noise and delay in the state estimate used for feedback control (see Figure 19).

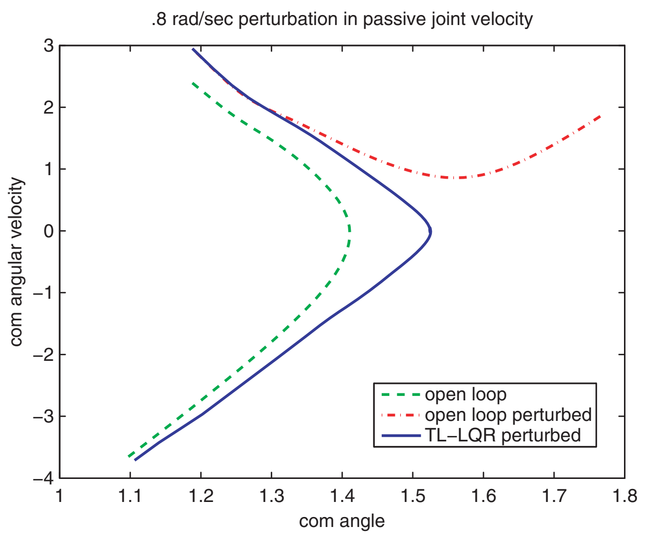

**Fig. 18.** Phase portrait of center-of-mass angle $\theta$ versus its derivative $\dot{\theta}$ for (1) a nominal half-bound trajectory, (2) open-loop execution with perturbation of +0.8 rad s$^{-1}$ in the initial condition, and (3) transverse-LQR stabilized trajectory with perturbed initial state.

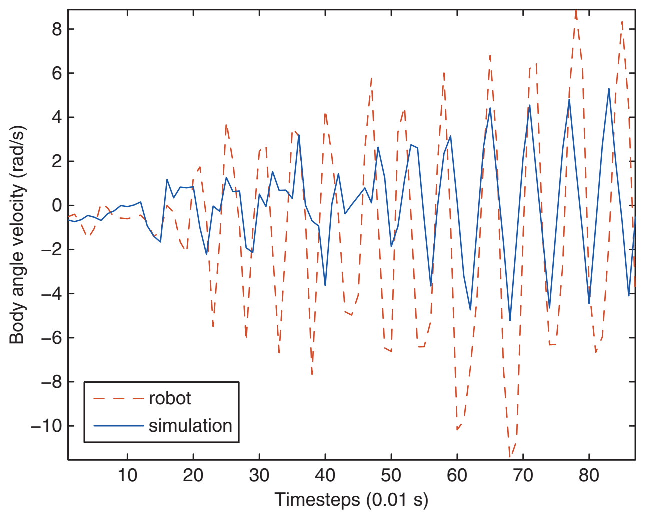

**Fig. 19.** Oscillations induced by measurement noise and delays: simulation and experiment.

We believe that if we can obtain better state estimates with less noise and reduced delay, we will be able to achieve stable bounding on the real robot, but this remains to be tried in future work.

## 6. Concluding Remarks and Future Directions

In this paper, we have demonstrated motion planning for the LittleDog quadruped robot to achieve bounding on very rough terrain. The robot was modeled as a planar system with a 16-dimensional state space. A physics-based simulation was developed and identified using data from the real robot. An efficient RRT-based planner was implemented. This planner used motion primitives to reduce the size of the action space, and also to produce smooth trajectories without an optimization step. Task-space guidance was implemented by sampling from a five-dimensional subset of the state space. The reduction in sampling dimension ignores pitch, vertical position, and velocities which significantly improves search efficiency. To handle challenging dynamic constraints associated with bounding, we used reachability guidance, so that random samples were chosen such that they were closer to reachable regions of the tree, rather than to the tree itself. Doing so increases the likelihood that the expansion step of the RRT algorithm can make progress in the direction of the sample, which dramatically improves RRT performance. The RRT motion planner was demonstrated to achieve bounding over steps and over logs, with terrain height differences corresponding to approximately 50% of the leg length. Without reachability guidance, 12 hours was not sufficient to plan in the otherwise identical conditions. Using task-space biasing and reachability guidance, we were able to plan within minutes. A primary reason for this is that a standard RRT implementation does not look ahead. Many of the nodes on the outer edges of the tree (the exact nodes which the Voronoi bias selects most often, because they are near the unexplored regions of state space) correspond to nodes that are either incapable of bounding on the next step because not enough energy is carried over, or have too much energy and no possible action could be applied on the next step to keep the robot from falling over. Expanding on these nodes is futile, but the standard RRT will try to expand them repeatedly causing the algorithm to fail to find a plan.

The motion primitives used for the motion planner were tested on the robot, and shown to be capable of bounding on rough terrain. The trajectories implemented on the robot using the motion primitives are naturally smooth, and our model captures the resulting dynamics quite well for short trajectories. However, the forward simulation and the actual behavior of the robot tended to diverge after about 1 second, despite having a relatively elaborate physics model of the robot that was identified on data collected from the robot. This is not surprising since the robot trajectories from similar initial conditions can diverge when executing the same open-loop trajectories. To address this, a transverse-LQR feedback controller was developed to stabilize the motion plan returned by the RRT. Unlike standard time-varying LQR methods, the transverse LQR makes no attempt to force the robot to converge to a trajectory in time, but rather forces it to converge to the trajectory as a path through state space, i.e. it is orbitally stabilizing. This makes it substantially more robust than regular LQR for underactuated systems. This type of control was shown to handle moderate disturbances as well as small delays in simulation. Significant delay and noise in the state estimate were shown to be destabilizing in simulation, producing similar behavior to what is observed on the real robot. In the future, we plan to address these issues by implementing better state estimation techniques, including a model-based observer, and continue the development of improved feedback control strategies.

In this work, the robot was assumed to be planar in order to reduce the state dimension for planning. As future work, the constraint that left and right feet move as one may be relaxed in order to extend this work to the 3D world where terrain heights are not even on both feet. The feet may be commanded different heights in order to conform to the terrain, while the planner assumes the foot position to be the average of the left and right foot positions. It would also be interesting to try to stabilize yaw and roll by asymmetrically changing the foot height on the right and left feet while bounding. Another direction for improving the planner would be to better characterize more of the reachable set, perhaps by incorporating knowledge about nearby terrain. Lastly, we believe the proposed motion planning approach can be generalized, and applied to a variety of other systems. In the near future, we expect to try a similar planning algorithm on a dynamic biped to achieve walking over rough terrain, and on a forklift operating in a highly constrained environment.

## Acknowledgements

This work was supported by the DARPA Learning Locomotion program (AFRL contract # FA8650-05-C-7262). The authors would also like to thank Sara Itani for her help during Phase III, especially in tuning an open-loop bounding gate.

## References

Altendorfer R, Moore N, Komsuoglu H, Buehler M, Brown HB, McMordie D, et al. (2001) RHex: A biologically inspired hexapod runner. *Autonomous Robots* 11: 207–213.

Berges P and Bowling A (2006) Rebound, slip, and compliance in the modeling and analysis of discrete impacts in legged locomotion. *Journal of Vibration and Control* 12: 1407–1430.

Byl K, Shkolnik A, Prentice S, Roy N and Tedrake R (2008) Reliable dynamic motions for a stiff quadruped. In *Proceedings of the 11th International Symposium on Experimental Robotics (ISER)*, Athens, Greece.

Byl K and Tedrake R (2009) Dynamically diverse legged locomotion for rough terrain. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan.

Chan N, Kuffner J and Zucker M (2008) Improved motion planning speed and safety using regions of inevitable collision. In *17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control (RoManSy'08)*, Tokyo, Japan.

Cheng P (2005) *Sampling-based Motion Planning with Differential Constraints*. Ph.D. thesis, University of Illinois at Urbana-Champaign, IL.

Chestnutt J, Lau M, Cheung KM, Kuffner J, Hodgins JK and Kanade T (2005) Footstep planning for the honda asimo humanoid. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 05)*, Barcelona.

Chevallereau C, Abba G, Aoustin Y, Plestan F, Westervelt ER, Canudas-De-Wit C, et al. (2003). Rabbit: a testbed for advanced control theory. *IEEE Control Systems Magazine* 23: 57–79.

Collins SH, Ruina A, Tedrake R and Wisse M (2005) Efficient bipedal robots based on passive-dynamic walkers. *Science* 307: 1082–1085.

Fraichard T (2007) A short paper about motion safety. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 07)*, Rome, Italy, pp. 1140–1145.

Fraichard T and Asama H (2003) Inevitable collision states. a step towards safer robots? In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 03)*, Volume 1, Las Vegas, NV, pp. 388–393.

Frazzoli E, Dahleh M and Feron E (2005) Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics* 21: 1077–1091.

Frazzoli E, Dahleh MA and Feron E (2002) Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics* 25: 116–129.

Gilardi G and Sharf I (2002) Literature survey of contact dynamics modelling. *Mechanism and Machine Theory* 37:1213–1239.

Hahn W (1967) *Stability of Motion*. Berlin: Springer-Verlag.

Hale JK (1980) *Ordinary Differential Equations*. New York: Robert E. Krieger Publishing Company.

Harada K, Hattori S, Hirukawa H, Morisawa M, Kajita S and Yoshida E (2007) Motion planning for walking pattern generation of humanoid. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 07)*, pp. 4227–4233.

Hauser J and Chung CC (1994) Converse Lyapunov functions for exponentially stable periodic orbits. *Systems and Control Letters* 23: 27–34.

Hauser K (2008) *Motion Planning for Legged and Humanoid Robots*. Ph.D. thesis, Stanford.

Hauser K, Bretl T, Latombe J-C, Harada K and Wilcox B (2008) Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research* 27: 1325–1349.

Hirose M and Ogawa K (2007) Honda humanoid robots development. *Philosophical Transactions of the Royal Society* 365: 11–19.

Hunt KH and Crossley FRE (1975) Coefficient of restitution interpreted as damping in vibroimpact. *Journal of Applied Mechanics Series E* 42: 440–445.

Kajita S, Kanehiro F, Kaneko K, Fujiware K, Harada K, Yokoi K, et al. (2003) Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 03)*, Taipei, Taiwan, pp. 1620–1626.

Kaneko K, Kanehiro F, Kajita S, Hirukawa H, Kawasaki T, Hirata M, et al. (2004) Humanoid robot HRP-2. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 04)*, New Orleans, LA, pp. 1083–1090.

Kavraki L, Svestka P, Latombe J and Overmars M (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4): 566–580.

Kolter JZ, Rodgers MP and Ng AY (2008) A control architecture for quadruped locomotion over rough terrain. In *Proceedings* *of the IEEE/RAS International Conference on Robotics and Automation (ICRA 08)*, Pasadena, CA, pp. 811–818.

Kuffner JJ, Kagami S, Nishiwaki K, Inaba M and Inoue H (2002) Dynamically-stable motion planning for humanoid robots. *Autonomous Robots* 12: 105–118.

Kuffner JJ, Nishiwaki K, Kagami S, Inaba M and Inoue H (2003) Motion planning for humanoid robots. In Dario P and Chatila R (Eds), *ISRR* (*Springer Tracts in Advanced Robotics*, volume 15). Berlin: Springer, pp. 365–374.

LaValle S and Branicky M (2002) On the relationship between classical grid search and probabilistic roadmaps. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, Nice, France.

LaValle SM (2006) *Planning Algorithms*. Cambridge: Cambridge University Press.

LaValle SM and Kuffner JJ (2001) Randomized kinodynamic planning. *The International Journal of Robotics Research* 20: 378–400.

Manchester, IR (2010) Transverse dynamics and regions of stability for nonlinear hybrid limit cycles. arXiv:1010.2241 [math.OC].

Manchester IR, Mettin U, Iida F and Tedrake R (2009) Stable dynamic walking over rough terrain: Theory and experiment. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, Lucerne, Switzerland.

Marhefka D and Orin D (1996) Simulation of contact using a nonlinear damping model. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 96)*, Vol. 2, Minneapolis, MN, pp. 1662–1668.

Pongas D, Mistry M and Schaal S (2007) A robust quadruped walking gait for traversing rough terrain. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 07)*.

Pratt JE and Tedrake R (2005) Velocity based stability margins for fast bipedal walking. In *Proceedings of the First Ruperto Carola Symposium on Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*, Vol. 340, pp. 299–324.

Raibert M, Blankespoor K, Nelson G, Playter R and the BigDog Team (2008) Bigdog, the rough-terrain quadruped robot. *Proceedings of the 17th World Congress of the International Federation of Automatic Control*.

Raibert M, Chepponis M and Brown H (1986) Running on four legs as though they were one. *IEEE Journal of Robotics and Automation* 2: 70–82.

Raibert MH (1986) *Legged Robots That Balance*. Cambridge, MA: The MIT Press.

Ratliff N, Zucker M, Bagnell JAD and Srinivasa S (2009) Chomp: Gradient optimization techniques for efficient motion planning. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan.

Rebula J, Neuhaus P, Bonnlander B, Johnson M and Pratt J (2007) A controller for the littledog quadruped walking on rough terrain. *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 07)*.

Sakagami Y, Watanabe R, Aoyama C, Matsunaga S and Fujimura NHK (2002) The intelligent ASIMO: system overview and integration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 02)*, Vol. 3, Switzerland, pp. 2478–2483.

Shiriaev AS, Freidovich LB and Manchester IR (2008) Can we make a robot ballerina perform a pirouette? Orbital stabilization of periodic motions of underactuated mechanical systems. *Annual Reviews in Control* 32: 200–211.

Shkolnik A and Tedrake R (2009) Path planning in 1000+ dimensions using a task-space Voronoi bias. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan.

Shkolnik A, Walter M and Tedrake R (2009) Reachability-guided sampling for planning under differential constraints. In *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, pp. 2859–2865.

Song G and Zefran M (2006) Stabilization of hybrid periodic orbits with application to bipedal walking. In *Proceedings of the 2006 American Control Conference*, Minneapolis, MN.

Sugihara T (2004) *Mobility Enhancement Control of Humanoid Robot based on Reaction Force Manipulation via Whole Body Motion*. Ph.D. thesis, University of Tokyo.

Westervelt ER, Grizzle JW, Chevallereau C, Choi JH and Morris B (2007) *Feedback Control of Dynamic Bipedal Robot Locomotion*. Boca Raton, FL: CRC Press.

Westervelt ER, Grizzle JW and Koditschek DE (2003) Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control* 48: 42–56.

Yershova A, Jaillet L, Simeon T and LaValle S (2005) Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA 05)*, pp. 3856–3861.

Zucker M (2009) *A Data-Driven Approach to High Level Planning*. PhD thesis, Carnegie Mellon University, available at http://www.cs.cmu.edu/ mzucker/mz-thesis-proposal.pdf.

## Appendix A: Ground Interaction Model

This appendix gives on overview of the computation of ground contact forces for the LittleDog model.

Ground contact models can be discrete or continuous (see Gilardi and Sharf (2002) for an overview). Discrete collision modeling can range from using a constant coefficient of elasticity to more advanced approaches that can predict slipping behavior and the presence or absence of bounce (Berges and Bowling 2006). Discrete modeling is advantageous because of its simplicity, but is not well suited to LittleDog, because it assumes an instantaneous change in momentum, whereas on the robot compression of shin springs extends the collision duration to a time scale comparable with the rest of LittleDog dynamics. Continuous impact modeling is more suited for LittleDog and can be subdivided into modeling the forces normal and tangential to the surface.

The continuous ground contact model presented here carefully computes the interaction of LittleDog feet with rough terrain, allowing it to predict shin-spring displacement, foot roll, foot slip, compliance and energy dissipation during ground collision, and bounce when too little energy is dissipated.

### A.1. Terrain Model and Foot Roll

The feet on LittleDog are small rubber balls about 2 cm in diameter. When the angle of the leg to the terrain changes, the ball rolls, producing a noticeable displacement. This is equivalent to a movement of the ball's center along a different terrain, which is offset from the original by the foot radius. To account for this effect, given a terrain height map, $\gamma(x)$, a new terrain height map, $\gamma^*(x)$, is computed such that every point on it is exactly one ball radius, $r_b$, away from the original terrain:

$$\begin{aligned} r_b &= \min_z \left\{ (\gamma^*(x) - \gamma(z))^2 + (x - z)^2 \right\}, \forall x \\ \gamma^*(x) &> \gamma(x), \forall x \end{aligned} \tag{8}$$

Both height maps can be seen in Figure 3, where the bottom light blue horizontal line is $\gamma(x)$, the original terrain, and the black line above it is $\gamma^*(x)$, the terrain computed by Equation (8). In this case, $\gamma^*(x) = \gamma(x) + r_b$, but this is not generally true for non-flat terrain.

When a LittleDog foot rolls, the velocity of the foot center is different from the velocity at the ground contact by $r_b\dot{\theta}$, where $\dot{\theta}$ is the absolute angular velocity of the foot. The new height map and the adjustment to ground contact velocity completely capture the foot roll behavior.

All ground contact computations use the new height map, $\gamma^*(x)$, referred to as "the terrain" below. A function for the slope of the terrain, $\alpha(x)$, is computed from $\gamma^*(x)$.

## *A.2. Ground Friction Model*

The friction force between the ground and the feet is assumed to be a smooth function of velocity and to be tangent to the ground surface:

$$\frac{F_f}{N} = K_f \arctan(K_d \dot{s}). \tag{9}$$

Here, $F_f$ is the friction force, $N$ is the surface normal force, $\dot{s}$ is the ground contact velocity, and $K_f$ and $K_d$ are model parameters.

To fit the data, the robot was commanded to hold its legs straight and placed on an inclined surface. The steady-state velocity as well as the normal and tangential forces were measured for a variety of slopes and are shown in Figure 20, along with a fit of the friction model.

For high magnitudes of velocity, the friction force equation resembles that of Coulomb friction. As the magnitude decreases, the force drops off to smoothly change direction at zero velocity. The smoothness of the function is important for integration purposes and seems to be a good approximation except for extremely small velocities. At small velocities, the friction coefficient is small and might produce drift, but it is negligible in typical timescales of a simulation run (less than 1 minute). An arc-tangent was selected for the functional form because it fits the available data well, but other sigmoids could be used.

## *A.3. Ground Forces Computation*

A diagram of a foot in collision with flat terrain is shown in Figure 21. The figure shows a portion of the robot shin in

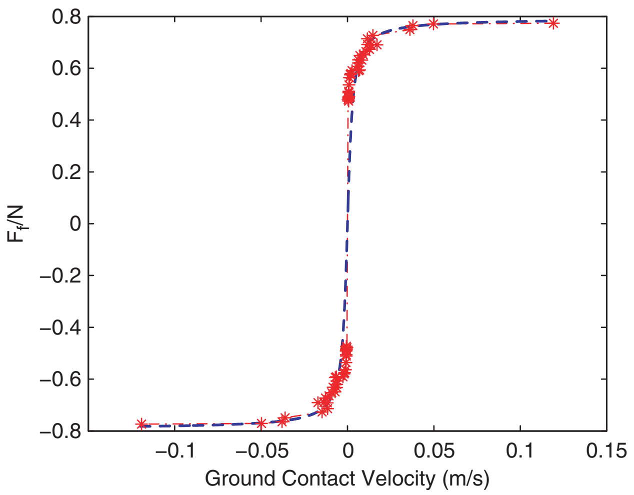

**Fig. 20.** Friction coefficient versus steady-state velocity on an inclined plane.

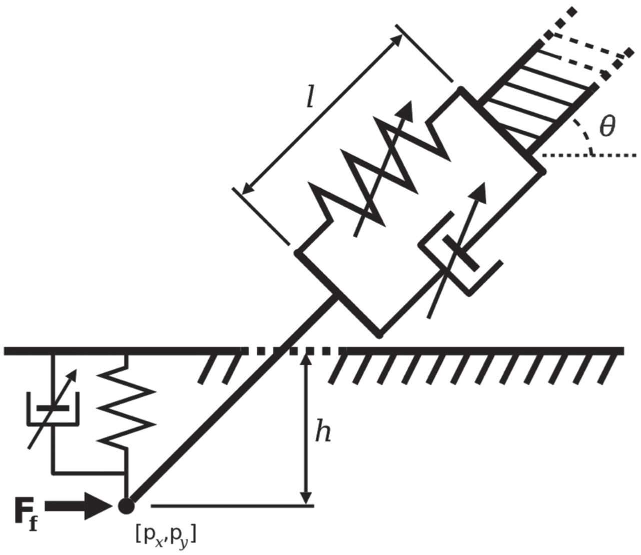

**Fig. 21.** Ground contact model. A foot with a center at $[p_x, p_y]$ is attached to a shin spring of length $l$ at an angle of $\theta$ to the terrain. The foot penetrates a distance $h$ into the ground, which is modeled as a compressible plane. A velocity-dependent friction force is applied at the point of ground contact. The terrain angle at the ground contact point is equal to $\alpha$, and is not shown in the figure.

the top right, connected to the shin spring of length $l$, which is modeled as a non-linear spring damper. Connected to the shin spring is the foot, the center of which is shown below the ground in the figure at position $[p_x, p_y]$. The center of each foot is computed from the current state of the robot. Whenever a foot center $[p_x, p_y]$ is below the terrain, $p_y < \gamma^*(p_x)$, the foot is considered to be in collision.

The foot below the shin spring is assumed to be massless. Therefore, the sum of forces acting on it is zero and is given by

$$\vec{F}_f + \vec{N} + \vec{M} + \vec{P} = 0, \tag{10}$$

where $F_f$ is the friction force, $N$ is the normal force, $M$ is the shin-spring force, and $P$ is the perpendicular force applied on the foot by the spring housing.

The normal force model uses a non-linear spring damper of the form introduced by Hunt and Crossley (1975):

$$N = K_h h(1 - \zeta_h \dot{h}), \tag{11}$$

where $h$ is the penetration depth, $\dot{h}$ is the rate of change of the penetration, and $K_h$ and $\zeta_h$ are constants. Compared with linear damping, it has the advantage of being continuous across the ground contact and avoiding sticking forces between surfaces for almost all cases (Marhefka and Orin 1996). The penetration depth, $h$, is computed as the shortest distance between the foot center and the height map, $\gamma^*(x)$, and is perpendicular to the height map. Since on the actual robot the foot cannot overlap the terrain, it is assumed that any overlap is due to compliance in the leg, the rubber foot, or the ground.

Note that $\dot{h}$ is an algebraic function of $[\dot{p_x}, \dot{p_y}]$, the foot center velocity, which in turn is an algebraic function of the known robot state and $\dot{l}$. Unlike $l$, $\dot{l}$ is not a part of the state, so $\dot{h}$ cannot be computed directly. The normal force is affine in $\dot{l}$, so, for a robot in state $x$, Equation (11) can be rewritten as

$$N = N_x(x) + N_l(x)\dot{l}, \tag{12}$$

where $N_x(x)$ an $N_l(x)$ are non-linear functions of the state.

The actual shin spring on the robot is limited in its range of travel. During a bounding motion, it is typical for the spring to reach the limits of motion, where it hits a hard stop. The spring is modeled as linear in its normal range and to have a hard collision at the travel limits of the same functional form as the normal force. Assuming a rest length of $l_0$, the displacement from rest is $\delta l = l - l_0$, and the range of travel for $\delta l$ is between 0 and $l_{\max}$, the force is given by

$$M = \begin{cases} K_s \delta l + b_s \dot{l} + K_c \delta l(1 + \zeta_l \dot{l}), & \delta l < 0 \\ K_s \delta l + b_s \dot{l}, & 0 \le \delta l < l_{\max} \\ K_s \delta l + b_s \dot{l} + K_c(\delta l - l_{\max})(1 - \zeta_l \dot{l}), & l_{\max} \le \delta l, \end{cases} \tag{13}$$

where $K_s$ and $K_c \gg K_s$ are stiffness parameters, and $b_s$ and $\zeta_l$ are damping parameters. Similarly to the normal force, the spring force is affine in $\dot{l}$ and can be written as

$$M = M_x(x) + M_l(x)\dot{l} \tag{14}$$

for some non-linear functions of the state $M_x(x)$ and $M_l(x)$.

The friction force is given by Equation (9), where $\dot{s}$ is the velocity of the foot center along the height map $\gamma^*(x)$ and can be computed from the state of the robot $x$, the slope of the terrain at the ground contact $\alpha$, and $\dot{l}$.

The force applied to the foot by the spring housing, $P$, is unknown, but can be eliminated from the force balance by only considering the component of Equation (10) that is orthogonal to $P$. Noting that $M \perp P$, $F_f \perp N$, the angle between the spring and the ground is $\theta - \alpha$, and substituting (9) into (10) gives

$$\begin{aligned} 0 &= F_f \cos(\theta - \alpha) + N(x,\dot{l})\sin(\theta - \alpha) - M(x,\dot{l}) \\ &= [K_f \arctan(K_d \dot{s}(x,\dot{l})\cos(\theta - \alpha)) + \sin(\theta - \alpha)] \\ &\quad N(x,\dot{l}) - M(x,\dot{l}), \end{aligned} \tag{15}$$

which is just a function of the robot state and $\dot{l}$, and where $N(x,\dot{l})$and $M(x,\dot{l})$ are given by Equations (11) and (14), respectively.

By approximating the arc-tangent function, Equation (15) is used to find $\dot{l}$, which is then used to find the normal and friction forces using Equations (11) and (9). The forces are then applied to the appropriate point of the rigid five-link model and $\dot{l}$ is used to update the shin-spring length. Although for some parameters and system states, Equation (15) might have multiple solutions for $\dot{l}$, in practice, the $\dot{l}$ with the lower magnitude can be chosen as the physically plausible solution. All of the ground interaction forces are dissipative, so the dynamics are guaranteed to remain stable.

For a foot not in collision, no forces are applied on the foot, so $F_f = N = 0 \rightarrow M = 0$. The rate of change of the shin length is then, from Equation (14),

$$\dot{l} = -\frac{M_l(x)}{M_x(x)}, \tag{16}$$

which is a fast, stable non-linear system that drives $l$ to $l_0$ quickly after the foot leaves the ground.

## Appendix B: Model Parameters

The motor model was assumed to be independent of the other parameters and fit to real joint trajectory data. The fit accurately predicts the behavior of the joints, as seen in Figure 4, which shows the model performance on a different trajectory. The friction coefficients in the ground force model were fit to steady-state sliding as described in Appendix A. The rest of the ground contact model was identified by commanding the robot to hold its legs straight down, parallel to each other, dropping it vertically onto flat terrain, and fitting the parameters to the resulting body trajectory.

The total mass of the robot, the lengths of each link, and the maximum shin spring travel were measured directly. The remaining parameters, including inertias of the links, the mass distribution between the links, and COM locations, were fit to a large number of short bounding trajectories. The cost function for the fit was a quadratic form on the distance between actual and simulated feet positions, which captures the effect of the three unactuated variables ($x$, $y$, and body pitch), neglecting the unactuated shin springs that are not considered to be a part of the configuration.

All of the fits were computed with non-linear function optimization (using MATLAB's fminsearch). In total, 34 parameters were fit for the model. Table 1 lists the parameters and their values.

**Table 1.** LittleDog model parameters.

| Symbol | Value | Units | Description |
|---|---|---|---|
| **Rigid-body model** | | | |
| Shin | | | |
| $m_1$ | 0.13 | kg | Mass of shin |
| $l_1$ | 9.2 | cm | Length of shin |
| $I_1$ | $7 \times 10^{-5}$ | kg m$^2$ | Inertia of shin |
| $cx_1$ | 5.7 | cm | Center of mass in $x$ for shin |
| $cy_1$ | 0.3 | cm | Center of mass in $y$ for shin |
| Upper leg | | | |
| $m_2$ | 0.24 | kg | Mass of upper leg |
| $l_2$ | 7.5 | cm | Length of upper leg |
| $I_2$ | $6 \times 10^{-6}$ | kg m$^2$ | Inertia of upper leg |
| $cx_2$ | 4.8 | cm | Center of mass in $x$ for upper leg |
| $cy_2$ | −0.9 | cm | Center of mass in $y$ for upper leg |
| Main body | | | |
| $m_3$ | 2.3 | kg | Mass of body |
| $l_3$ | 20.2 | cm | Length of body |
| $I_3$ | $2.2 \times 10^{-3}$ | kg m$^2$ | Inertia of body |
| $cx_3$ | 8.7 | cm | Center of mass in $x$ for body |
| $cy_3$ | −0.2 | cm | Center of mass in $y$ for body |
| **Motor model** | | | |
| Hip joint | | | |
| $k_{\text{hip}}$ | 3,800 | s$^{-2}$ | Hip gain |
| $b_{\text{hip}}$ | 98 | s$^{-1}$ | Hip damping |
| $\bar{v}_{\text{hip}}$ | 7.9 | rad s$^{-1}$ | Hip velocity saturation |
| $\bar{a}_{\text{hip}}$ | 200 | rad s$^{-2}$ | Hip acceleration saturation |
| Knee joint | | | |
| $k_{\text{knee}}$ | 9,700 | s$^{-2}$ | Knee gain |
| $b_{\text{knee}}$ | 148 | s$^{-1}$ | Knee damping |
| $\bar{v}_{\text{knee}}$ | 12 | rad s$^{-1}$ | Knee velocity saturation |
| $\bar{a}_{\text{knee}}$ | 430 | rad s$^{-2}$ | Knee acceleration saturation |
| **Ground contact model** | | | |
| Friction | | | |
| $K_f$ | 0.5 | — | Friction gain |
| $K_d$ | 1,000 | s m$^{-1}$ | Friction velocity slope |
| Normal spring | | | |
| $K_h$ | $1.4 \times 10^5$ | N m$^{-1}$ | Ground stiffness |
| $\zeta_h$ | 1.4 | s m$^{-1}$ | Ground damping |
| Shin spring | | | |
| $K_s$ | 7,500 | N m$^{-1}$ | Spring linear stiffness |
| $b_s$ | 180 | Ns m$^{-1}$ | Spring linear damping |
| $K_c$ | 7,500 | N m$^{-1}$ | Spring limit stiffness |
| $\zeta_l$ | 180 | s m$^{-1}$ | Spring limit damping |
| $l_{\max}$ | 0.88 | cm | Maximum spring travel (spring limit) |
| $\beta$ | 0.29 | rad | Angle between spring and shin joint |
| $r_b$ | 1.0 | cm | Foot radius |

For the rigid body model, the parameters are heavily coupled and some of the individual values might not be accurate. This is true of the inertias and to some degree of the centers of masses. Because the planar model lumps two LittleDog legs into a single leg, the leg masses and inertias, as well as some of the stiffnesses and damping values, are twice as large as their physical counterparts for a single leg.

The length of the shin is given from the knee joint to the foot center, assuming full extension of the shin spring. Centers of masses are given in the reference frames of their links, with $x$ pointing along its link and $y$ perpendicular to it in a right-handed convention. The origin of the back shin is at the back foot and $\hat{x}$ points toward the knee joint, the origin of the back upper leg is at the knee joint and its $\hat{x}$ points toward the hip joint, and the origin of the body is at the back hip joint, with its $\hat{x}$ pointing toward the front hip joint. The front links have mirror symmetry with the back legs.

## Appendix C: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org

**Table 2.** Table of Multimedia Extensions

| Extension | Type | Description |
|---|---|---|
| 1 | Video | Simulation results: demonstration of motion plans over rough terrain, and feedback stabilization |