Motion Planning for Bounding on Rough Terrain with the LittleDog Robot

Alexander Shkolnik, Michael Levashov, Sara Itani and Russ Tedrake

Abstract— **In this paper we develop an RRT-based motion planner that achieved bounding in simulation with the LittleDog robot over extremely rough terrain. LittleDog is a quadruped robot that has 12 actuators, and a 36-dimensional state space; the task of bounding involves differential contstraints due to underactuation and motor limits, which makes motion planning extremely challenging. Rapidly-exploring Random Trees (RRTs) are well known for fast kinematic path planning in high dimensional configuration spaces in the presence of obstacles, but the performance of the basic RRT algorithm rapidly degrades with addition of differential constraints and increasing dimensionality. To speed up the planning we modified the basic RRT algorithm by (1) biasing the search in task space, (2) implementing the Reachability-Guided RRT, which dynamically changes the sampling region, and (3) by implementing a motion primitive which reduces the dimensionality of the problem. With these modifications, the planning algorithm succesfully generated plans over very rough terrain. Short trajectories were demonstrated to work open-loop on a real robot.**

I. Introduction

Agile locomotion over rough terrain seems easy for animals, but has proven very challenging for robots. Reliable mobility within human environments is currently a limiting factor preventing robots from being utilized more in society. In this work we present progress towards achieving agile locomotion over rough terrain using the LittleDog robot. LittleDog is a small 3kg position controlled quadruped robot with point feet and was developed by Boston Dynamics under the DARPA Learning Locomotion program. Our task is to navigate known uneven terrain, as quickly as possible. An A* based motion planning algorithm can be used to generate a statically-stable gait for walking over rough terrain. To speed up the motions and make the robot capable of traversing more difficult terrain, dynamic maneuvers are necessary. Such maneuvers consist of lifting several feet at the same time. While improving gait speed, this behavior eliminates the support polygon, which makes planning and control more difficult. We have previously demonstrated the ability to introduce short stereotyped dynamic maneuvers into a gait, while relying on the intermittent existence of a support polygon to regain control between motions [1]. Doing more agile maneuvers continuously over rough terrain is intuitively what an animal would do, but is a challenging motion planning problem on many levels.

A. Shkolnik is a PhD candidate in EECS at the Computer Science and Artificial Intelligence Lab, MIT, 32 Vassar St., Cambridge, MA 02139, USA shkolnik@mit.edu

M. Levashov is a PhD candidate in the dept. of Aeronautics And Astronautics, MIT, 32 Vassar St., Cambridge, MA 02139, USA levashov@mit.edu

S. Itani is an undergraduate in EECS, MIT, 32 Vassar St., Cambridge, MA 02139, USA levashov@mit.edu

R. Tedrake is an Assistant Professor of EECS at the Computer Science and Artificial Intelligence Lab, MIT, 32 Vassar St., Cambridge, MA 02139, USA russt@mit.edu

The LittleDog robot, like many robots used in walking research, is moderately stiff, designed to accurately position its joints (limbs). The robot uses a high gear ratio, making the joints non-backdriveable, and has limited compliant elements. The large gear ratio enables an onboard PD controller to generate precise movements with little regard for applied torques. Although these types of robots have fairly good position accuracy, the design trade-offs include a limit on the maximum speed of motion and inability to transfer between kinetic and potential energy via compliant mechanisms. These limitations present a significant challenge in achieving highly dynamic maneuvers over rough terrain where position accuracy is also required. In this work we utilize task-space biasing and reachability-guidance to implement an RRT-type motion planning framework to quickly find feasible bounding motion plans, in which the robot alternates between supporting itself with the hind legs and the front legs. The RRT respects the kinodynamic constraints, and the generated plans can then be locally optimized and executed on the robot with feedback stabilization. The focus of this paper is on the motion planning component itself.

The remainder of this paper is organized as follows: 1) we begin by reviewing background information, including alternative approaches to achieve legged locomotion; 2) a framework for bounding is presented, including discussion of a physics based model of the LittleDog robot and various constraints that are imposed; 3) we review the concept of Reachability Guidance (RG) to improve RRT search speed for dynamic systems; 4) this is followed by discussion of a simple motion primitive for bounding, which fits neatly into the RG framework; 5) we then describe a sampling strategy based on the motion primitive which is of reduced dimension so as to produce a Task Space bias; 6) a fast method for approximating the Reachable set is presented; 7) Results are described, which demonstrate fast bounding motion planning in simulation over extremely rough terrain, and feasibility of the approach is explored by experiments using the motion primitive executed open loop on the LittleDog robot hardware; 8) we conclude with discussion.

II. Background

The problem of fast locomotion over rough terrain has long been a research topic in robotics, beginning with the seminal work by Raibert in the 1980's [2], [3]. Many of the past approaches have utilized compliant or mechanically clever designs that enable passive stability, or reflexive control algorithms that tend to react to terrain in an attempt to enforce stability with "local feedback". Such methods do not take into account knowledge about upcoming terrain. Kinodynamic planning algorithms have also made significant headway, but have been used mostly for kinematic path planning in configuration space, focusing on maneuvers requiring dexterity and obstacle avoidance, but where velocities are not important for the task. Such an approach excludes agile locomotion that relies on dynamic gaits, which is sensitive to configurations and velocities. In this work we attempt to bridge this gap, by demonstrating a framework for fast global motion planning that respects kinodynamic constraints. This is a necessary step to enable agile locomotion on a position controlled robot, and would be complemented by local trajectory optimization, feedback stabilization, a good model of the dynamics as well as good sensing of the state and environment. The benefit of a model based approach is that inherent limitations of the system, which are present in many robots, can be taken into full consideration when determining a motion plan, enabling a generalized controller.

Work in legged robotics can be approximately classified into three categories: 1) passively stabilizing, e.g. through clever mechanical design, 2) actively stabilizing by reflexively adjusting a gait, and 3) locomotion with careful foot placement.

In the first category, in one extreme, there are passive dynamic walkers that require little or no control input or even actuators in order to walk if provided with a flat, slightly downward sloping surface [4]. Many other robot systems utilize open loop walking or bounding gaits, and through the coupling between the mechanical system and careful gait design, attain desired behavior even with small disturbances in the terrain. For example, quadruped robots with elastic elements in the joints can bound using very simple frequency or gait-based control algorithms thanks to the mechanical design [5]. Other approaches utilize adaptive control together with similar designs with elastic elements to constantly tune CPG or harmonic based gait parameters online, e.g. [6]–[9]. These approaches can achieve moderate speed, and are stable over mildly varying terrain, but are reactive, and do not preempt the gait with knowledge of upcoming terrain. Another very successful example is Rhex [10], which was developed around the concept of anchoring a hexapod robot to a simplified dynamic model that can be easily controlled, but also relies on clever mechanical design for stabilization on rougher terrain and is not capable of careful foot placement. The Raibert / Leg Lab hopper robots also rely on simplification of the problem, for example by relying on an aerial phase of the gait, and assuming that (near) massless legs can move into landing position without impacting the dynamics of the rest of the system [2]. This approach has worked well, and research continues on the BigDog platform [11], but requires powerful (typically hydraulically driven) actuators.

Many robots utilize high frequency feedback for stabilization. Many of the position controlled bipeds use it for ZMP stabilization [12], [13], while BigDog uses it for pitch and roll stabilization. TVLQR or LQR based on transverse linearization [14] can also be used. However, with the exception of the biped community, most of the approaches discussed so far utilize reactive feedback for stability, and do little in terms of careful foot placement which will be required for traversing rougher terrain. The Learning Locomotion project, utilizing the LittleDog robot, has pushed the envelope on walking control using careful foot placement. Much of this work has combined path planning and motion primitives to enable crawling gaits on rough terrain e.g. [15], [16].

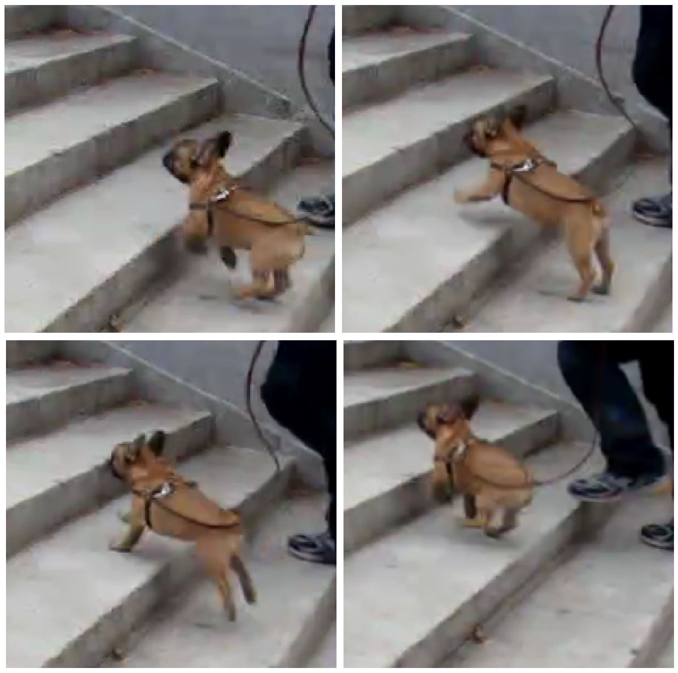


Fig. 1. Dog bounding up stairs.
Images from video at http://www.flickr.com/photos/istolethetv/3321159490/

III. Framework

We use a five-link planar rigid-body model of LittleDog, with continuous dynamics obtained by system identification, with discrete-time control decisions made at 100Hz. The planar model assumes that the back legs move together as one and the front legs move together as one. For control purposes, we define the dynamics as

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

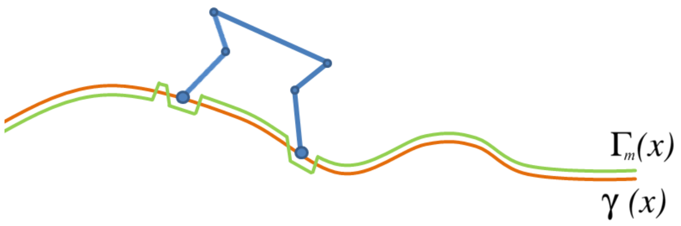


Fig. 2. Sketch of a virtual obstacle function, $\Gamma_m(x)$ in relation to the ground, $\gamma(x)$.

where $\mathbf{x} \in \mathcal{X} = [\mathbf{q}, \dot{\mathbf{q}}]$, $\mathbf{q}$ is the 7 dimensional (two positions, pitch, and 4 joint angles) configuration space. f comes from a fixed-step 4th order Runge-Kutta integration of the continuous model fit from our system identification procedure. A stateless ground model was implemented, assuming a spring in the leg, e.g. a spring mounted approximately parallel to the lower leg located at the foot, in addition to a stiffer vertical spring in the ground itself. A friction model controlled horizontal sliding. In total there were 35 parameters of the system which were fit.

Given an initial condition $\mathbf{x}_0$, a goal center of mass location, x_G, and a ground height function $\gamma(x)$, we formulate the trajectory design problem as:

$$\min_{\mathbf{u}[1],\ldots\mathbf{u}[n_f]} \quad \sum_{n=1}^{n_f} (\mathbf{u}[n] - \mathbf{q_a}[n])^T \mathbf{R} (\mathbf{u}[n] - \mathbf{q_a}[n]),$$
$$\text{subject to}$$
$$\mathbf{x}(0) = \mathbf{x}_0,$$
$$\text{com}_x(\mathbf{x}[n_f]) \geq x_G,$$
$$\forall i \in [1,...,4], \forall n \quad |\dot{q}_{a,i}[n]| \leq \text{vlim}_i,$$
$$\forall i \in [1,...,4], \forall n \quad |\ddot{q}_{a,i}(q_{a,i}[n], \dot{q}_{a,i}[n], u_i[n])| \leq \text{alim}_i,$$
$$\forall i \in [1,...,4], \forall n \quad |\tau_i(\mathbf{x}[n], \mathbf{u}[n])| \leq \text{taulim}_i,$$
$$\forall l \in [1,2], \forall n \quad \text{footpos}_z(\mathbf{x}[n], l) \geq \Gamma_m(\text{footpos}_x(\mathbf{x}[n], l),$$
$$where\, m_{n+1} \in [m_n, m_n + 1] = S(m_n, x_n) \qquad (2)$$

where $vlim_i$, $alim_i$ and $taulim_i$ are the i'th motor velocity acceleration and torque limits. $\mathbf{q}_a$ is the position vector of each actuated joint, and therefore $(\mathbf{u} - \mathbf{q}_a)$ is correlated with the motor torque. The functions $z = \Gamma_m(x)$ are virtual obstacle functions, an example of which is sketched in Figure 2. The enumerator, m, is the current sub-motion being performed, and is kept track of by a state machine S. Sub-motions in the context of this paper are based on motion primitives which will be described in depth below. In this paper, the state machine simply increments when a foot touches the ground between bounding motions. The $\Gamma_m(x)$ function shown above provides a virtual constraint which ensures that feet during stance do not slip much, and requires the front foot to achieve some clearance over the ground when moving it forward. The function $\Gamma_{m+1}(x)$ would prevent the front foot from slipping, while ensuring the back foot clears the terrain briefly when making forward progress. In this paper, the Γ virtual obstacle functions are defined by the feet positions of the robot at the time when the state machine increments. An alternative strategy, not pursued further in this work, is to utilize a higher level foothold planner, and predefine the Γ functions around these footholds.

IV. Reachability-Guided RRT Overview

Random 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 samples to nodes in the tree. The Euclidean distance metric is easy to implement, and works well for holonomic path planning problems, but this metric breaks down when constraints imposed by the governing equations of motion restrict the direction that nodes can grow to. To deal with this, we developed a modified version of the RRT algorithm called the Reachability-Guided RRT (RG-RRT) [17]. Reachability-guidance 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 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, and therefore such a sample-node pair is not useful for exploration, and is discarded. On the other hand, if the sample is nearest to a reachable region of the tree, the parent node corresponding to the 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 one. 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. Discrete actions can be explicitly determined and stored as discrete reachable states. Or, a continuous motion primitive action space can also be approximated by a discretization of the action space, resulting in a discrete set of reachable nodes which might be quite far in state space from their parent node. This idea suggests the RG-RRT also naturally extends to models with hybrid dynamics, simply by 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, $\Delta t \in [\Delta T_{low}, \Delta T_{high}]$, according to the state equations (1) and the set of available control inputs, $\mathcal{U}$.

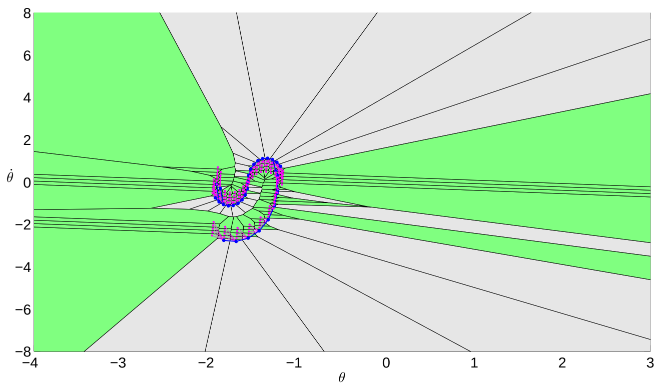
The structure of the RG-RRT algorithm is outlined in Algorithm 1. Given an initial point in state space, x_{init}, the first step is to add the state as a node in the tree. Given the point x_{init}, the InsertNode() function solves for the set of reachable points in the state space that are consistent with the differential constraints (1). For many systems of interest, the approximate bounds of the reachable

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

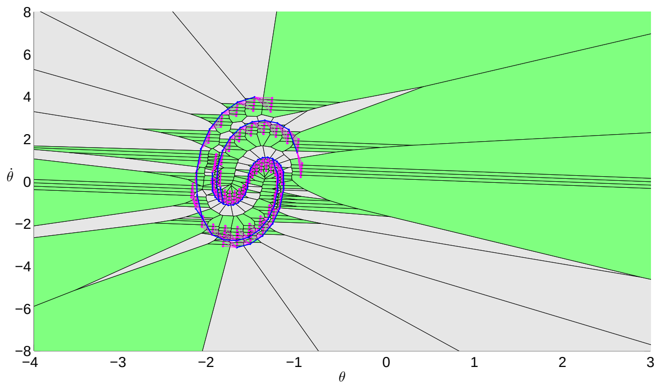
```
1:  $\mathcal{T} \leftarrow$ INITIALIZETREE();
2:  $\mathcal{T} \leftarrow$ INSERTNODE($x_{\text{init}}, \mathcal{T}$);
3:  for $k = 1$ to $k = K$ do
4:      $x_{\text{rand}} \leftarrow$ RANDOMSTATE();
5:      $(x_{\text{near}}, x^r_{\text{near}}) \leftarrow$ NEARESTSTATE($x_{\text{rand}}, \mathcal{T}$);
6:      while $x_{\text{near}} = \{\}$ do
7:          $x_{\text{rand}} \leftarrow$ RANDOMSTATE();
8:          $(x_{\text{near}}, x^r_{\text{near}}) \leftarrow$ NEARESTSTATE($x_{\text{rand}}, \mathcal{T}$);
9:      end while
10:     $u \leftarrow$ SOLVEINPUT($x_{\text{near}}, x^r_{\text{near}}, x_{\text{rand}}, \mathcal{T}$);
11:     $x_{\text{new}} \leftarrow$ NEWSTATE($x_{\text{near}}, u$);
12:     $\mathcal{T} \leftarrow$ INSERTNODE($x_{\text{new}}, \mathcal{T}$);
13: end for
14: return $\mathcal{T}$
```

set can be generated by discretizing the action space, and then integrating the corresponding dynamics forward using any favored integration method. In some cases, with a short enough Δt, taking actions between the limits results in an integrated state that lies between the states produced when taking the actions at their limits. In such cases it is sufficient to consider only the bounds of the action set. The reachable set may also be approximated or learned. When the dimension of the action space increases, in particular, it may become more efficient to approximate the reachable set with a simple geometric function, for example an ellipsoid, if such a function can approximate the actual reachable volume. One benefit of generating the reachable set using discretized actions is that points that comprise the reachable set can also be efficiently tested for collisions before they are added to the Reachable set, in order to reduce the likelihood of trajectories leaving free space as part of the exploration phase.

With the node and its corresponding reachable set added to the tree, we draw a random sample, x_{rand}, in state space, typically from a uniform distribution, and use it to grow the tree. The NEARESTSTATE($x_{\text{rand}}, \mathcal{T}$) function compares the distance from the random sample not only to the nodes, but also to the points within their reachable sets. If the closest Reachable point is closer to the sample than the closest node of the tree, then both this reachable point, x^r_{near}, and its corresponding parent node, x_{near}, are returned. Otherwise, if the closest node of the tree is nearer to the sample than any Reachable point, the function returns an empty point pair, in which case the RG-RRT throws this sample away, and draws a new sample from the state space and repeats the process. As demonstrated in Figure 3, 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.



(a) 30 nodes



(b) 60 nodes

Fig. 3. RG-RRT Voronoi diagrams for a pendulum. The blue diagram in (a) corresponds to the tree 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 the tree. Green regions are areas where samples are 'allowed', and correspond to Voronoi areas associated with the reachable set. Samples that fall in any grey areas are discarded. Note that these regions are constantly changing as the tree grows.

V. MOTION PLANNER

A. *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 5 link planar arm with 4 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, ΔT. One drawback of this method is the resulting trajectory found by the RRT will 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 be sufficiently smooth. Another drawback of using a constant time step is that in order to ensure completeness with such an action space, ΔT should be relatively small (for LittleDog bounding, we found that .1 seconds seems to be appropriate). Empirically, however, the search time increases approximately in proportion to $1/\Delta T$, 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. Thus, a desired trajectory over ΔT can be generated using a smooth function $\mathcal{G} : [\mathbf{x}, \mathbf{u}, \Delta T] \mapsto \mathcal{X}_{[0,\Delta T]}$ constrained to have the initial position and velocity of the start node, $\mathbf{x}$, and a given end position and velocity specified by $\mathbf{u}$. This action set requires specifying two numbers for each actuated DOF, one for position, one for 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.

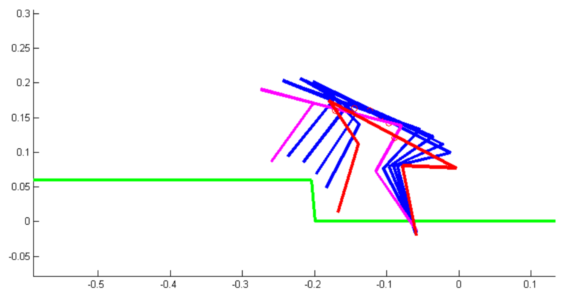


Fig. 4. First half of a double bound. Initial pose shown in red, final pose shown in magenta. Axes are in meters.

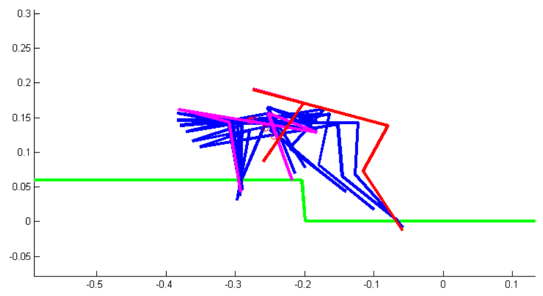


Fig. 5. Second half of a double bound. Initial pose shown in red, final pose shown in magenta. Notice that in both of these figures, the swing foot ”tucked in” in order to clear the step.

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, one can see that some simplifications can be made to the action space. This results in a much longer ΔT and, therefore, a shorter search time, while also producing smooth, jerk free 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 phase-1, the hind legs begin moving first, and the motion is always ”opening” the hip angles. This produces a forward acceleration of the COM, which ideally also generates a rotational moment around the pseudo pin joint around the hind foot. 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 starting pose to the ending pose in a straight line. The front feet move from the start to the end pose, however, because these feet are not weight bearing, it is useful to ”tuck” the feet while moving them in order to help avoid obstacles. Once the hind legs and front legs have reached their desired landing pose, the remaining trajectory is held constant until the front legs impact the ground. Examples of such a trajectory are shown in Figure 4 and the top image in Figure 6.

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”. The front leg begins the movement just before impact with the ground, and the back leg may be delayed slightly. The resulting motions are shown in Figure 5 and the second image in Figure 6.

Note that the start and end velocities in this motion primitive are always zero, a factor which reduces the action space further. Using these motion primitives requires a variable time step, ΔT, because this directly influences accelerations and, therefore, moments around the passive joints. However, for each phase, one only needs to specify 4-DOF, corresponding to the end pose of the system at the end of the phase.

B. *Sampling in Task Space*

In the RRT algorithm, sampling is typically performed uniformly over the Configuration Space, and actions are taken either to move directly towards the sample, or actions may be sampled uniformly over the Action Space. Such sampling creates a Voronoi Bias for fast exploration by often selecting nodes of the tree near unexplored regions, while occasionally refining within explored regions. We have previously shown that the Voronoi Bias can exist in the Configuration (State) Space, or it can be implemented in a lower dimensional Task Space [18]. Reducing the dimensionality of the search with a Task Space Voronoi Bias can significantly improve search efficiency and, if done carefully, does not impact 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. Additionally, 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 4-dimensional manifold. A sample is drawn by first choosing an x position of the robot, and then selecting 4 joint angles from the manifold, while checking to ensure that collision constraints are validated.

Given a sample described above, the y-coordinate of the foot is set to the ground position at x, and the passive joint is computed by the constraint that both feet are on the ground. Thus, sampling the 5 dimensional space maps to a point $\mathbf{q}_s$ in the 7 dimensional configuration space of the robot. When a sample is created, 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 is found which is closest to the Reachable region. An action, $\mathbf{u}$, is then created by

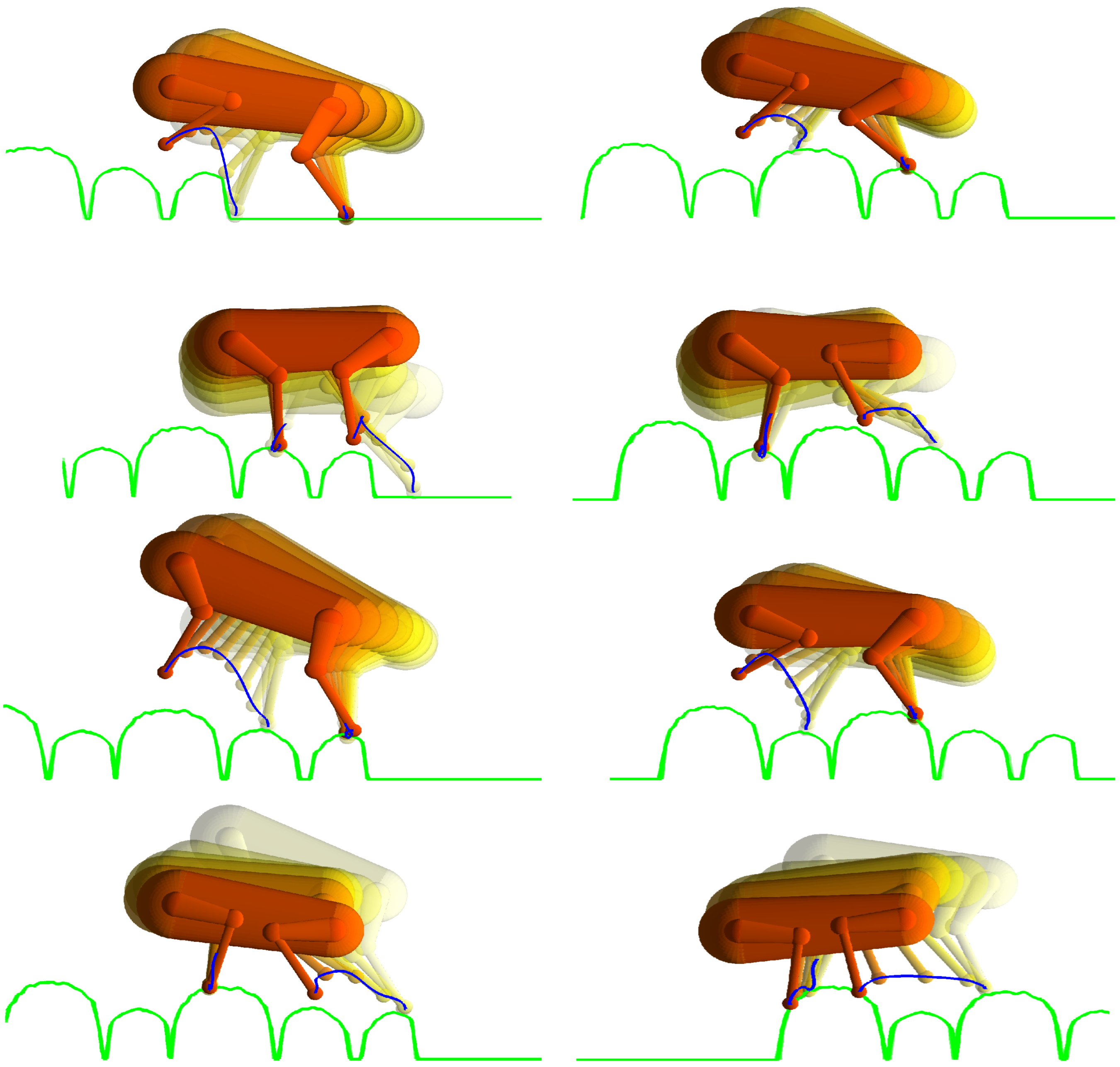


Fig. 6. Bounding trajectory over logs

selecting the 4 actuated joint angles from the sample, $\mathbf{q}_s$. An action time interval Δ_T is chosen by uniformly sampling from $T \in [.4, .7]$ seconds.

C. Generating the Reachable Set

We have demonstrated that Reachability Guidance can deal with issues imposed by dynamic constraints in systems such as the pendulum and acrobot [17]. In those systems, the action of the single actuator can be discretized, and the reachable set was approximated by interpolating along the resulting states achieved by applying the discrete set

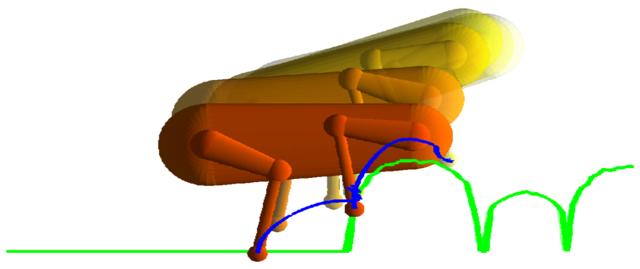


Fig. 7. Bounding trajectory over logs, continued

of actions. For the work described here, however, even the reduced 4-dimensional action space becomes too large to discretize efficiently. For example, discretizing with only 3 actions per dimension, there would be 81 actions to apply and simulate in order to approximate the reachable set for each node.

Instead of discretizing in joint space, we found that the reachable set can be better 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 flips 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 center of mass, and converts the kinetic energy into potential energy. On 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 these two extremes. Feasible actions between these extremes are usually rich enough to produce the desired bounding behavior, if it is possible for a given node.

VI. Results

The planning algorithm described in this paper was implemented to successfully find bounding trajectories up a series of 7cm 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 8cm. An example of a bounding trajectory is shown in Figures 6 and 7. The bottom link in the robot leg is 10cm, and the top link is 7.5cm; if we take into account that the bottom of the body is 3cm below the hip, 7cm represents approximately 50% of maximum leg travel of the robot, which corresponds to extremely rough terrain.

Experimenting with the real robot, we found, as expected, that openloop execution of the motion plan found by the RRT diverges with time from model predictions. Trajectories are typically 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 our motion primitives in our planning system we used the motion primitives defined in this paper to find a short bound sequence on the log terrain. Given that, unlike our model, the logs are not planar, we laid tracks corresponding to the stance width of the robot along the terrain and constrained the system to only allow foot holds where the points on adjacent tracks were of the same height. With some tuning, we were able to find some actions that produced bounding over the logs on the real robot, shown in Figure 8, using the motion primitives described above. This trajectory was successful approximately 20% of the time, even though we took care to have the same initial position for each trial.

Trajectory optimization and feedback stabilization will be required to enable the execution of the complete trajectory. Feedback will help to compensate for model errors and the inherent instability of the system itself, as demonstrated by our experiments on the actual robot. We took great care to ensure that our physics model is smooth and continuous, so gradients can be used to help in these tasks, but the actual implementation of a feedback controller is left to future work.

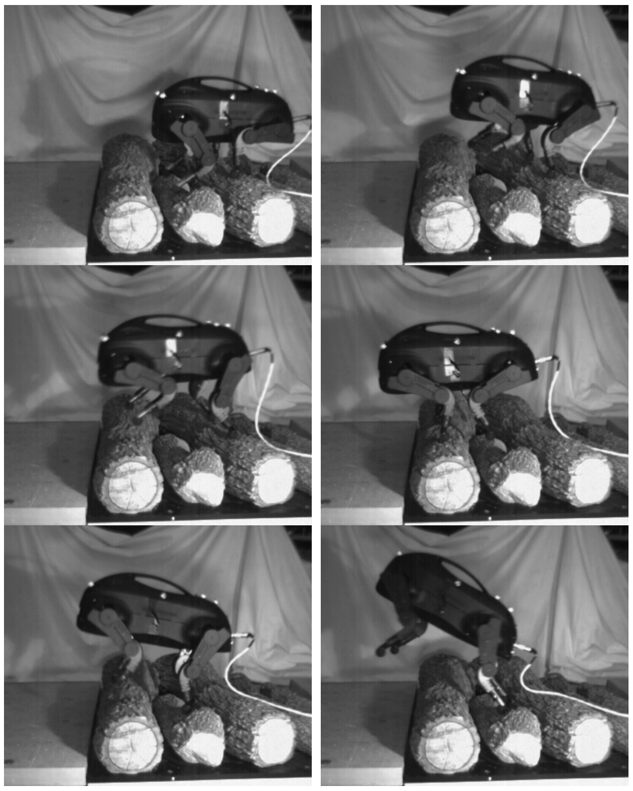


Fig. 8. Bounding over logs with LittleDog

VII. Concluding Remarks and Future Directions

In this paper we have demonstrated motion planning in simulation of the LittleDog quadruped robot to achieve bounding on very rough terrain. The robot has a 36 dimensional state space, which was reduced to a 14 dimensional state space in a planar model. A physics based model 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. Corresponding to the idea of Task-space guided RRT search, we sampled over a 5-dimensional manifold, with an injective mapping to the full 14 dimensional state space, so that the Euclidean distance metric could be applied efficiently. 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 than the tree itself. Doing so ensures that the expansion step of the RRT algorithm can make progress in the direction of the sample, and this vastly 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 Reachabiliy 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 edges of the tree – the exact nodes which the Voronoi Bias most often selects because they are near 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, including the logs terrain board that we were able to plan on. The trajectories implemented on the robot using the motion primitives are 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. Future work can focus on constructing a feedback controller that stabilizes the motion plan produced by the algorithm disclosed in this paper.

ACKNOWLEDGMENT

This work was supported by the DARPA Learning Locomotion program (AFRL contract # FA8650-05-C-7262).

REFERENCES

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

[2] M. H. Raibert, *Legged Robots That Balance*. The MIT Press, 1986.

[3] Raibert, M. H., Chepponis, M., Brown, and H. B., "Running on four legs as though they were one," *IEEE Journal of Robotics and Automation*, vol. 2, no. 2, pp. 70–82, 1986.

[4] S. H. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, pp. 1082–1085, February 18 2005.

[5] F. Iida, G. Gomez, and R. Pfeifer, "Exploiting body dynamics for controlling a running quadruped robot," in *Proceedings of the 12th International Conference on Advanced Robotics (ICAR)*, July 2005, pp. 229–235.

[6] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *International Journal of Robotics Research*, vol. 22, no. 3-4, pp. 187–202, March-April 2003.

[7] J. Buchli, F. Iida, and A. J. Ijspeert, "Finding resonance: Adaptive frequency oscillators for dynamic legged locomotion," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 06)*, pp. 3903 – 3909, 2006.

[8] Buchli, Jonas, Ijspeert, and A. Jan, "Self-organized adaptive legged locomotion in a compliant quadruped robot," *Auton. Robots*, vol. 25, no. 4, pp. 331–347, 2008.

[9] M. Ahmadi and M. Buehler, "A control strategy for stable passive running." IEEE Int. Conf. Intelligent Robots and Systems, 1995, pp. 152–157.

[10] R. Altendorfer, U. Saranli, H. Komsoglu, D. Koditschek, H. B. Brown, M. Buehler, N. Moore, D. McMordie, and R. Full, "Evidence for spring loaded inverted pendulum running in a hexapod robot," in *Proceedings of the 7th International Symposium on Experimental Robotics (ISER)*, 2000.

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

[12] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiware, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *ICRA IEEE International Conference on Robotics and Automation*. IEEE, Sep 2003, pp. 1620–1626.

[13] M. Hirose and K. Ogawa, "Honda humanoid robots development," *Philosophical Transactions of the Royal Society*, vol. 365, no. 1850, pp. 11–19, Jan 2007.

[14] I. R. Manchester, U. Mettin, F. Iida, and R. Tedrake, "Stable dynamic walking over rough terrain: Theory and experiment," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2009.

[15] J. Rebula, P. Neuhaus, B. Bonnlander, M. Johnson, and J. Pratt, "A controller for the littledog quadruped walking on rough terrain," *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy*, 2007.

[16] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 811–818.

[17] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA)*. IEEE/RAS, 2009.

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