# A Control Architecture for Quadruped Locomotion Over Rough Terrain

J. Zico Kolter, Mike P. Rodgers, and Andrew Y. Ng
Computer Science Department, Stanford University, Stanford, CA 94305

***Abstract*— Legged robots have the potential to navigate a much larger variety of terrain than their wheeled counterparts. In this paper we present a hierarchical control architecture that enables a quadruped, the "LittleDog" robot, to walk over rough terrain. The controller consists of a high-level planner that plans a set of footsteps across the terrain, a low-level planner that plans trajectories for the robot's feet and center of gravity (COG), and a low-level controller that tracks these desired trajectories using a set of closed-loop mechanisms. We conduct extensive experiments to verify that the controller is able to robustly cross a wide variety of challenging terrains, climbing over obstacles nearly as tall as the robot's legs. In addition, we highlight several elements of the controller that we found to be particularly crucial for robust locomotion, and which are applicable to quadruped robots in general. In such cases we conduct empirical evaluations to test the usefulness of these elements.**

## I. Introduction

Although wheeled robots are very fuel efficient, they are extremely limited in the types of terrain that they can reliably navigate. Legged robots, in contrast, offer the potential to navigate a much wider variety of terrain, as evidenced by the fact that biological legged animals are capable of accessing nearly all of the earth's land surface. This potential has sparked a great deal of research on legged locomotion in recent years, both for quadruped and biped robots. However, despite a great number of advances in the field, legged robots still lag far behind the capabilities of their biological cousins.

In this paper we consider the task of quadruped locomotion over challenging, irregular terrain, with obstacles nearly as tall as the robot's legs. We present a full control system that enables a quadruped robot known as "LittleDog," shown in Figure 1, to robustly navigate a wide variety of difficult terrains using a static walk. This work extends previous research by considering terrain that is significantly more challenging (relative to the size of the robot) than any previously published work of which we are aware.

While the overall performance of our system naturally depends on several factors (including some, such as the mechanical design of the robot, that are out of our control), throughout our work we have found that a few key elements of the controller and planner have a large impact on performance. In particular, the specific method for planning the robot's center of gravity (COG) trajectory and the use of closed-loop recovery and stabilization drastically improved performance of our system. These elements are applicable to quadruped locomotion in general, and we therefore describe them in detail and experimentally document their usefulness.

Fig. 1. The LittleDog robot, designed and built by Boston Dynamics, Inc.

The rest of this paper is organized as follows. In Section II we discuss related work. In Section III we present the full hierarchical control system for quadruped locomotion. Finally, in Section IV we present experimental results, and conclude the paper in Section V.

## II. Background and Related Work

While a complete survey of all quadruped locomotion literature is beyond the scope of this paper, we present a broad overview of the general themes present in this research. One of the fundamental distinctions in this literature is between *static* and *dynamic* gaits. Static gaits, such as a walk (or "crawl"), maintain static stability, which in the ideal setting means that the robot's center of mass is always within the polygon formed by its supporting legs. Dynamic gaits, such as a trot or gallop, do not have this requirement; although this allows for much faster locomotion, it comes at the cost of a much more difficult balancing task. Due to the challenges present in navigating highly irregular terrain, with obstacles nearly the size of the robots legs, we focus in this paper on statically stable gaits.

Statically stable gaits were first considered in the robotics literature by McGhee and Frank [1]. Since then, there have been a large number of proposed approaches to static gaits, [2], [3], [4], including many which are capable of walking on irregular terrain [5], [6], [7], though in these works the sizes of the irregularities in the terrain are typically much smaller than the size of the legs. Lee et al. [8] present a static gait capable of navigating over large obstacles, though in that work the obstacles considered were all box-shaped, so the robot is able to step entirely on flat surfaces.

Another vein of research has focused on dynamic gaits

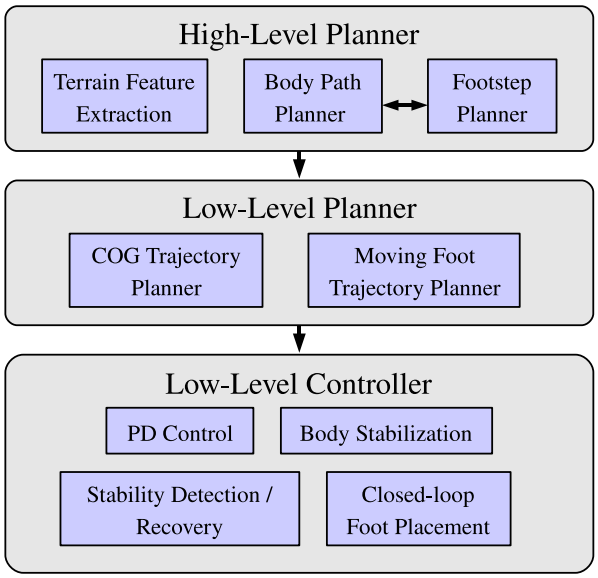



Fig. 2. Overview of the planning and control architecture.

where the robot "bounces" on compliant legs [9], [10], [11], [12]. While these gaits are capable of achieving very fast motion, they are highly limited in terms of the terrains they can reliably cross. There has also been work on adaptive gaits (both dynamic and static), that make use of a biologically inspired "central pattern general" to traverse rough terrain [13], [14], [15], [16], though again the terrain here typically has only very small irregularities compared to the size of the robot.

The work by Pongas et al. [17] and Rebula et al. [18] bears the most similarity to the current work. These both present systems for control of the LittleDog robot that share many similarities to the system we present here. We build upon this previous work in two ways. First, in this paper we consider terrain that is substantially more challenging than the terrain previously considered, at equal or faster speeds. Although this is somewhat of an unfair comparison (since these groups have more recently applied these techniques to more challenging terrain) based on the most recent public evaluations at the time of the original paper submission (September, 2007) our results reflect performance that was on par with the very best that had been achieved by any group working with the LittleDog robot. Second, although previous controllers have many elements similar to our own, the past work has only demonstrated performance for the entire system. In contrast, in this paper we conduct detailed experiments evaluating the empirical advantage of several of these elements individually.

## III. Planning and Control for Quadruped Locomotion

The full planning and control problem for a quadruped robot is to plan a sequence of joint angles that moves the robot to its desired position while maintaining stability, then apply control inputs (i.e., torques) to achieve this desired trajectory. However, due to the complexity of this task, we

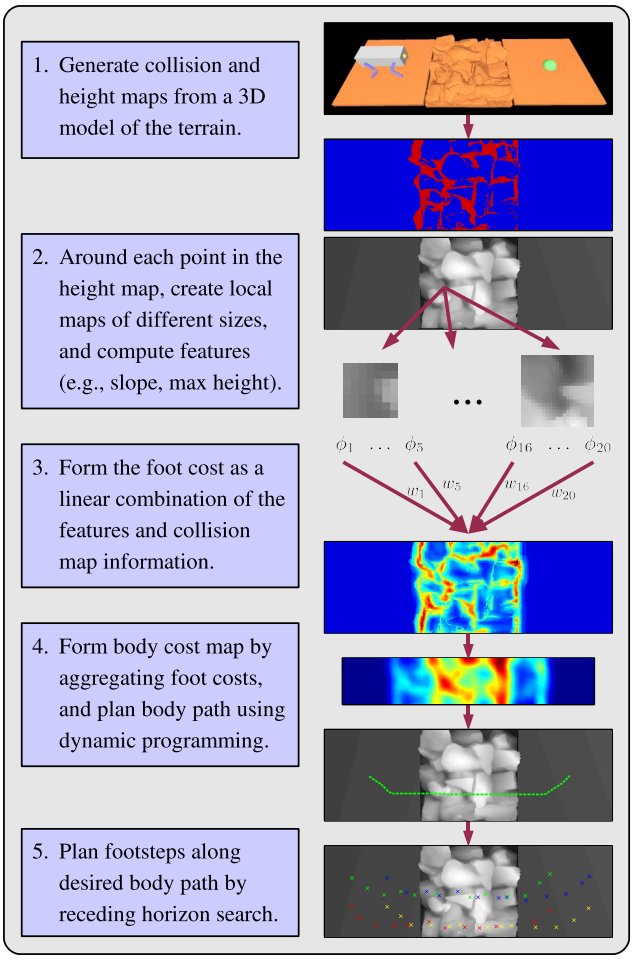



Fig. 3. Overview of the high-level planner.

make use of a hierarchical decomposition. Specifically, we separate the system into three components: 1) a high-level planner, which plans a series of footsteps across the terrain, 2) a low-level planner, which plans trajectories for the COG and feet so as to achieve the upcoming footsteps, and 3) a low-level controller, which provides control inputs that achieve the desired COG and feet trajectories in the face of disturbances. This architecture is shown in Figure 2. We now describe each element in detail, highlighting the differences between it and past approaches.

### A. The High-Level Planner

The goal of the high-level planner is to determine a set of feasible footsteps across the terrain, ideally one that is robust to minor deviations and slips of the robot. Given this objective, some foot locations are clearly superior to others; for example, stepping on flat surfaces or in small concavities is better than stepping on highly sloped surfaces. Therefore, the first step in the high-level planning is to build a "foot cost map" that indicates the desirability of stepping at any given point in the terrain. The footstep planning task is then to find the minimum cost set of footsteps across the terrain. However, due to the high-dimensionality of the space, even this search problem is difficult, so we again employ a

hierarchical decomposition. In particular, we average the foot costs around the default foot locations to form a "body cost map," then use dynamic programming to find the minimum-cost path. Finally, we plan a set of footsteps that approximately follows this path using a receding horizon search. An overview of this entire process is show in Figure 3 and we now describe each of these steps in greater detail.

**1. Generate height and collision maps of the terrain.** The terrain is described natively as a set of 3D tri-meshes along with their positions and orientations, so we begin the planning process by sampling the heights in a grid to produce a discrete height map. In addition, a crucial aspect of planning good footsteps is to ensure that they do not cause excessive collision with the terrain. For example, if the robot attempts to place a rear foot directly in front of a large step, this will most likely cause the knee to collide with the step, knocking the robot off its desired trajectory. To ensure that this does not occur, we precompute approximate collision maps; in simulation, we place the robot's feet at each point in the height map, and determine whether the "default" pose[1] will cause the legs to collide with the terrain.

**2. Generate local features of the terrain.** At each point in the grid, we consider local height maps of different sizes (squares of 5, 7, 11, and 21 grid cells around the current point), and generate five features for each of these maps:

1) standard deviation of the heights
2) average slope in the $x$ direction
3) average slope in the $y$ direction
4) maximum height relative to the center point
5) minimum height relative to the center point

We do this for each of the four local map sizes, leading to a total of 20 features that describe local characteristics of the terrain at different spatial scales. In addition, we also include a boolean feature that indicates whether or not the given position causes a collision, as specified by the collision map described above, and a constant offset. This leads to a total of 22 features, which we represent by the vector $\phi(x) \in \mathbb{R}^{22}$ (note that for a given point on the terrain we actually form four feature vectors, one corresponding to each foot, with local features properly reflected to account for symmetry of the robot).

**3. Generate foot cost maps.** Given a set of features describing each point of the terrain, we take a linear combination of these features to form a cost map representing the desirability of that location for each foot (i.e., the cost of a point $x$ becomes $w^T\phi(x)$ for some weight vector $w \in \mathbb{R}^{22}$).

Of course, a crucial element to this system is choosing a weight vector $w$ that produces the proper costs. A good choice of $w$ has to trade off several features, and it is highly non-trivial to simply tune the coefficients by hand. The algorithm we use to learn the coefficients is called Hierarchical Apprenticeship Learning (HAL), and it allows a "teacher" to demonstrate good actions at multiple levels of the control hierarchy [19]. Very briefly, the HAL algorithm requires that user specify good footsteps at a few key and good approximate paths for the body over the terrain. It then uses information from both these levels to learn a the weights of a cost function which can be used both for the body path and footstep planning levels. In practice, we have found that *demonstrating* good behavior in this way is far easier than hand-coding a set of rules that induces the behavior.

**4. Form body cost map and plan body path.** To form the body cost map, we aggregate the foot cost maps in a square around the default foot location for each of the four feet. While this is only an approximation, since the robot's feet could be in many other locations given the body position, it nonetheless does capture the "expected" cost incurred for a given body position, and therefore serves as a mechanism for planning the desired path for the robot's body. Given this body cost map, we then use value iteration, a dynamic programming algorithm, to plan a minimum-cost path across the terrain. Note that this body path is merely a tool for limiting the search space for our footstep planner; the low-level planner itself only looks at the footsteps and does not attempt to follow the body path exactly.

**5. Plan footsteps along the desired body path.** Given a desired path for the robot's body, the final step of the planning process is to plan a set of footsteps that (roughly) follow this path. We plan the footsteps sequentially — moving first the back-right foot, then front-right, then back-left, then front-left — following the standard biological gait pattern for static walking [1]. Starting at the robot's initial location, we move the robot's center some distance along the desired body path, then look for low-cost foot placement around the default foot location of the moving foot. Because this "greedy" placement might lead to suboptimal foot placements in the future, we use a receding horizon branching search to find the placement that leads to a low sum of the foot costs for several steps in advance. We require that each footstep obey kinematic feasibility and that it not cause collisions with the terrain.

The primary goal of the hierarchical decomposition for the quadruped task is to speed up planning time, and there has been previous work that uses beam-search [8] or an A* variant [20] to accomplish similar goals. The advantage of the hierarchical decomposition we propose is that it both works well in practice, and allows us to apply the previously mentioned method for hierarchical apprenticeship learning.

### *B. The Low-Level Planner*

The goal of the low-level planner is to plan a desired trajectory for the robot's COG and moving feet so as to achieve the upcoming footsteps while maintaining static stability. For reasons that will soon be apparent, we actually plan a trajectory for the upcoming *two* footsteps, starting with one of the hind feet. Note that this plan for the COG is not the same as the "body path plan" mentioned above in the high-

[1]The high-level planner frequently uses the notion of a *default* location for the feet, or (equivalently) a default pose for the robot. This is simply a $(x, y, z)$ location for the foot relative to the body that is "good" in the sense that it gives the robot a stable pose while still allowing for a fair amount of kinematic reachability. While this is often an approximation (because, for example, whether or not the foot collides with the terrain at a given location can depend on the precise pose of the robot), it nonetheless captures the "expected" behavior in many cases, and greatly reduces computation.

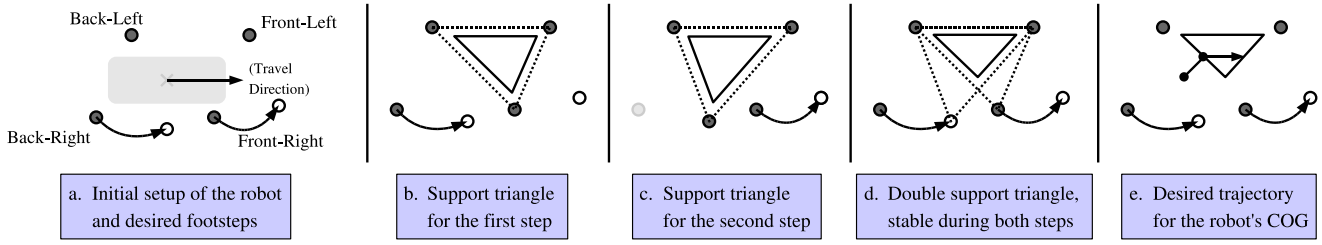



Fig. 4. Planning of the COG trajectory for a sequence of two steps.

level planner: the body path plan is just an approximate path for the robot over the terrain, used to speed up the footstep search, while the COG trajectory is the actual path that the COG should follow in order to maintain static stability.

Consider the robot pose shown in Figure 4(a), where the filled circles represent the current locations of the robot's feet, and the open circles represent the desired footsteps for the back-right and front-right feet. We begin by moving the back-right foot. When moving this foot, the robot will be statically stable if the COG is in the support triangle formed by the other three feet, as shown in Figure 4(b) — we take a standard approach, and inset the support triangle by some fixed margin, to be more robust to slight deviations or inaccuracies in the state estimation. Likewise, when moving the front-right foot, the robot will be statically stable if the COG is within the inset support triangle of the three new supporting legs, as shown in Figure 4(c). A key observation, noted in [1] among others, is that the supporting triangles for these two steps overlap, so that if the COG is moved properly, it is often possible to start lifting the front foot as soon as the back foot touches the ground. In contrast, when transitioning between moving the front-right foot and the back-left foot, the support triangles are disjoint, so some period of time must be spent shifting the COG with all four feet on the ground.

The guiding principle behind the our COG planning method is that we want to minimize the distance travelled by the COG while the robot is not moving its feet. To achieve this, we first compute the intersection of the two supporting triangles for each step, which we refer to as the *double* supporting triangle, shown in Figure 4(d). We then project the current COG into this double support triangle, and move it to this point. Because the robot's COG is not yet in the double support triangle during this "shifting" phase, it is not able to lift either foot and must instead keep all four feet on the ground. Finally, we move the COG from this projected location to its final location inside the double support triangle (which is the projection of the robot's effective center, the average of its four feet after taking the two footsteps, into the double support triangle). During this phase the robot can lift either foot, as the COG is in the support triangle for both moving feet; in our approach we move the COG half the distance while moving the back foot, then the remaining distance while moving the front foot. As mentioned, the advantage of this method is that we minimize the time spent moving the COG while the robot is not moving its feet, and we never move the COG backwards. These two effects allow for fast locomotion, even over very challenging terrain.

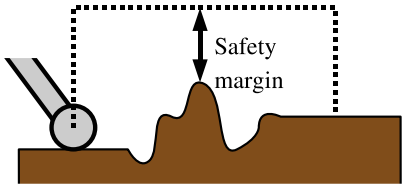



Fig. 5. Desired trajectory for moving feet over obstacles.

We determine the height and pitch of the robot's body by the height of the four feet. Given four foot locations, we set the the height to be the average height of these four foot locations, plus a constant amount. We determine the pitch by the average height of the front two feet relative to the average height of the back two feet. While moving a foot we interpolate between the height and pitch defined by the initial foot location and the height and pitch defined by the final foot location.

Finally, to plan the desired trajectories for the moving feet we take a very simple approach, and move the feet in a box pattern, where the height of the box is determined to lie some margin above the tallest obstacle crossed by the foot. This approach is shown in Figure 5. While this is not the shortest possible path, an advantage of this approach is that we do not need to adjust the foot trajectory based on *where* the tallest obstacle is relative to the foot's path.

Previous work on quadruped locomotion has often focused on periodic patterns (for example, sinusoidal trajectories) for COG movement, with a fixed *duty factor* (percentage of time spent moving the feet versus merely shifting the COG) [21], [2]. While such trajectories can be very efficient for fixed walking patterns, and have desirable properties such as smoothness (i.e., being twice differentiable) they are much more difficult to maintain given the irregular footsteps required to navigate difficult terrain. Recently, researchers have proposed a technique for online modification of periodic COG trajectories that maintains the smooth nature of these trajectories while adjusting for the current position of the feet, and applied this to the LittleDog robot [17]. However, in their implementation, the robot's COG still moves slightly backwards for a short time period during each step. As we will show in the next section, our system is able to achieve very robust performance on challenging terrain even with our non-smooth COG trajectories, and we therefore prefer to

minimize the length of these trajectories as much as possible, thereby increasing the speed of locomotion.

The COG trajectory planning used in [18] bears a great deal of similarity to our own, though they do not explicitly consider the double supporting triangle. When executing the irregular footstep patterns required to navigate large obstacles, we have found that often times the projection of the current COG into the first supporting triangle actually lies *ahead* of the projection of the final center into the second supporting triangle. By explicitly considering the double support triangle, we guarantee that we never need to move the COG backwards. In addition, the system presented in [18] occasionally opts to move the COG into the *center* of the current supporting triangle for a greater stability margin. While this technique can increase stability we found that for our system, moving the COG into the center of the supporting triangle actually often *increased* the chance of falling over. We discuss this effect further in Section IV.

### C. Low-Level Controller

Given the desired trajectories for the COG and moving feet, we use inverse kinematics to convert these to joint trajectories, then use a PD controller to apply torques that move the robot along these trajectories. However, due to the challenging nature of the terrains we consider, this approach alone is highly unreliable. Regardless of how well we plan, and regardless of how well the individual joints track their desired trajectories, it is almost inevitable that at some point the robot will slip slightly and deviate from its desired trajectory. Therefore, a critical element of our system is a set of closed-loop control mechanisms that detect failures and either stabilize the robot along its desired trajectory or re-plan entirely. In particular, we found three elements to be especially crucial: 1) stability detection and recovery, 2) body stabilization, and 3) closed-loop foot placement. We now describe each of these in greater detail.

**Stability Detection and Recovery.** Recall that (ignoring friction effects, which do not appear to have a major effect in the terrain we consider) the robot is statically stable only if the projection of the COG onto the ground plane lies within the triangle formed by the supporting feet (also projected onto the ground plane). If the robot slips while following its trajectory, the COG can move outside the supporting triangle, causing the robot to tip over. To counteract this effect, we compute the current (double) support triangle at each time step, based on the current locations of the feet as determined by state estimation. If the COG lies outside this triangle, then we re-run the low-level planner (planning only one step if the robot falls while moving a front foot). This has the effect of lowering all the robot's feet to the ground, then re-shifting the COG back into the inset support triangle.

**Body Stabilization.** While sometimes the recovery procedure in unavoidable, as much as possible we would like to ensure that the COG does *not* move outside the supporting triangle, even in light of minor slips. To accomplish this, we adjust the commanded positions of the supporting feet so as to direct the COG toward its desired trajectory. In particular, we multiply the commanded positions of the supporting feet by a transformation that will move the robot's COG from its current position and orientation to its desired position and orientation (assuming the supporting feet are fixed to the ground).

More formally, let $T_{des}$ be the $4\times4$ homogeneous transformation matrix specifying the *desired* position and orientation of the robot relative to the world frame, and similarly let $T_{cur}$ be the homogeneous transformation specifying the *current* position and orientation of the robot relative to the world frame. In addition, let $feet$ denote the default commanded positions of the supporting feet expressed in the robot's frame of reference, based on the desired trajectory for the COG. If we transform the commanded positions for the feet by

$$T_{des}^{-1}T_{cur}feet$$

then (assuming the supporting feet remain fixed) this would move the COG to its desired position and orientation. However, when coupled with PD control, this typically leads to large oscillations, so instead we employ a common interpolation scheme and command the supporting feet according to

$$(1-\alpha)feet + \alpha T_{des}^{-1}T_{cur}feet$$

for some $0 < \alpha < 1$ (in our experiments we found $\alpha = 0.1$ to be a good value). This causes the robot's COG to move gradually to track the desired trajectory, even if the robot slips slightly. In addition, we project the desired position $T_{des}$ into the current (double) supporting triangle, thereby working to stabilize the robot even if the initially computed trajectory becomes unstable due to the feet slipping. During our development we found this approach to be slightly more robust than attempting to move the supporting feet individually to stabilize the body, as our method keeps intact the relative positions of the supporting feet, leading to fewer unstable configurations. While the stabilization does lead to kinematic infeasibilites on occasion, we find that usually in such cases the robot is about to fall, and the recovery procedure shortly replans (feasible) footsteps.

**Closed-loop Foot Placement.** Finally, we want to ensure that the moving foot tracks its desired trajectory as closely as possible, even if the body deviates from its desired path. To accomplish this, at each time step we compute the desired location of the foot along its (global) trajectory, and use inverse kinematics based on the *current* pose of the robot's body to find a set of joint angles that achieves the desired foot location. This is particularly important in cases where the robot slips downward. If the robot's body is below its desired position and we merely execute an open loop trajectory for the moving foot, then the foot can punch into the ground, knocking the robot over faster than we can stabilize it. Computing a closed-loop trajectory for the foot in the manner described above avoids this situation.

It may seem as if there are also cases where closed-loop foot placement could actually hinder the robot rather than help. For example, if the robot is falling, then it may be best to simply put its foot down, rather than attempt to keep

TABLE I

THE FOUR TERRAINS USED FOR EVALUATION.

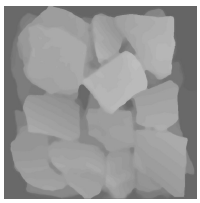
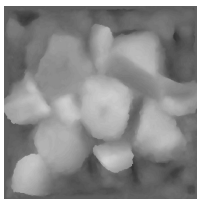
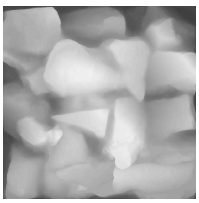
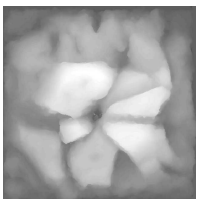
| **Terrain #** | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **Max Height** | 6.4 cm | 8.0 cm | 10.5 cm | 11.7 cm |
| **Picture** |  |  |  |  |
| **Heightmap** |  |  |  |  |


Fig. 6. Example of a typical setup for the robot and terrain (shown here with Terrain #3).

its foot along the proper (global) trajectory. However, in our experience this nearly always occurs in situations where the recovery procedure mentioned previously will catch the robot anyway, and this is borne out in the experimental results, as we will discuss shortly.

## IV. EXPERIMENTAL RESULTS

In this section we present experimental results for our controller on a variety of challenging terrain. The chief result is that our system is able to reliably cross difficult terrain at relatively fast speeds. In order to better understand the performance of the system, we perform experiments to analyze its behavior with and without several of the planning and control elements described above, thereby demonstrating their usefulness.

A video of the robot crossing the evaluation terrains is included with the paper. A higher-resolution version of this video is available at

`http://cs.stanford.edu/~kolter/icra08videos/`

### A. Experimental Setup

The LittleDog robot, shown in Figure 1, was designed and built by Boston Dynamics, Inc. It is a quadruped robot, about 30 cm long, whose legs lie 12 cm below its body when fully extended; the robot weights about 3kg. The robot has 12 independently actuated electric motors, three on each leg. A separate "host" computer performs nearly all processing, running a control loop at 100hz and relaying servo commands to the robot once every 10ms over a wireless channel. The robot's on-board hardware runs a joint-level PD servo controller at 500hz.

A motion capture (MOCAP) system estimates the position and orientation of the robot's body by tracking reflective markers attached to the robot. Joint encoders provide estimates of the robot's joint angles. Although we have experimented extensively with the use of additional localization methods, such as Kalman filters, we found in general that the raw estimates provided by the MOCAP and joint encoders were sufficient for complete pose estimation. The robot also contains an on-board IMU that can provide orientation estimates, but we do not make use of this component in our current system. Finally, the MOCAP also tracks markers on the terrains; we combine this with 3D models of the terrain to estimate the state of the entire environment.

We evaluated the performance of our system on four different terrains of varying difficulty, two that were provided by the official LittleDog program and two that we built ourselves. Pictures of the terrains and their corresponding heightmaps are shown in Table I. For all the terrains we considered all four crossing directions, and planned five separate paths across each direction by varying the initial position of the robot. This lead to a total of 20 paths across

TABLE II

SUCCESS PROBABILITIES OUT OF 20 RUNS ACROSS DIFFERENT TERRAINS FOR THE CONTROLLER WITH AND WITHOUT RECOVERY, BODY STABILIZATION, AND CLOSED-LOOP FOOT PLACEMENT.

| Terrain | All | w/o Rec. | w/o Stab. | w/o CLF | None |
|---|---|---|---|---|---|
| 1 | 100% | 100% | 100% | 100% | 100% |
| 2 | **100%** | 60% | 95% | 95% | 55% |
| 3 | **95%** | 25% | 55% | 75% | 35% |
| 4 | **95%** | 0% | 75% | 85% | 35% |
| Total | **97.5%** | 46.25% | 81.25% | 88.75% | 56.25% |

TABLE III

SUCCESS PROBABILITIES AND AVERAGE SPEED (IN CM/S) OF SUCCESSFUL RUNS FOR OUR COG TRAJECTORY PLANNING METHOD, FOR KEEPING THE COG FIXED WHILE MOVING THE FEET, AND FOR MOVING THE COG TO THE CENTER OF THE SUPPORTING TRIANGLE.

| Terrain | Our Method | Fixed COG | Centered COG |
|---|---|---|---|
| 1 | **100% (3.67)** | 100% (2.99) | 100% (2.28) |
| 2 | **100% (3.31)** | 75% (2.61) | 70% (2.22) |
| 3 | **95% (3.07)** | 60% (2.62) | 35% (2.10) |
| 4 | **95% (3.01)** | 50% (2.45) | 30 % (2.26) |
| Total | **97.5% (3.28)** | 71.25% (2.72) | 58.75% (2.23) |

each terrain. A standard setup for the robot and terrain is shown in Figure 6. Figure 7 shows several snapshots of the robot crossing Terrain #3.

### *B. Results and Discussion*

Overall, the system we present is able to successfully cross a wide variety of challenging terrains with a 97.5% success rate, at an average speed of 3.28 cm/sec. To put these results in context, we note that at the most recent public test of the LittleDog systems (which had an overall setup virtually identical to the one we use for our experiments, but which used a different terrain), the controller we describe in this paper achieved a speed of 3.63 cm/s on its best run, which was the fastest time of any group across this terrain; the next-fastest time on this test by another team was 2.85 cm/s. This result should be taken with some caution, since there are a very limited number of these tests, and not all teams were optimizing for speed. Nonetheless, based on these tests and personal communication with other researchers, we feel justified in our claim that the system we present is on par with the very best that had been achieved with the LittleDog robot at the time of original submission (September, 2007).

In our first set of experiments, we analyze the performance of the system with and without the low-level controller elements described previously. For each of the 20 planned paths across each of the four terrains, we evaluated the performance of our system with and without the stability detection and recovery, body stabilization, and closed-loop foot placement. In addition, we evaluated the performance of the system with none of these elements enabled. As shown in Table II, the controller with all elements enabled substantially outperforms the controller when disabling any of these three elements. This effect becomes more pronounced as the terrains become more difficult: Terrain #1 is easy enough that all the controllers achieve 100% sucess rates, but for Terrains #3 and #4, the advantage of using all the control elements is clear.[2]

Subjectively, the failure modes of the different controllers are as expected. Without the stability detection and recovery, the robot frequently falls over entirely after slipping a small amount. Without body stabilization, the robot becomes noticeably less stable during small slips, which sometimes leads to falls that even the recovery routine cannot salvage. Without closed-loop foot placement, the feet can punch into the ground during slips, occasionally flipping the robot. One interesting effect is that without recovery, the controller actually performs *worse* with body stabilization and closed loop foot movement enabled, especially on the more challenging terrains. This appears to be due to the fact that when the robot falls significantly (and makes no attempt to recover) both the body stabilization and closed-loop foot placement attempt to make large changes to the joint angles, causing the robot to become less stable. However, with recovery enabled the robot never strays too far from its desired trajectory without attempting to re-plan; in this case the advantage of using the body stabilization and closed-loop foot placement is clear from the experiments above.

In our second set of experiments, we compare the performance of the our COG trajectory planning method versus several alternatives. In particular, we consider a planner that does not make use of the double support triangle, but merely projects the COG into the current support triangle and keeps it fixed in this location while moving the feet. Second, we consider a planner that moves the COG into center of the supporting triangle as done (for some steps) in [18]. Table III shows the success percentages and speeds for each of the different methods. Not surprisingly, keeping the COG fixed or moving it to the center of the supporting triangle significantly lowers the speed of the the gait. More surprisingly, however, is the fact that for harder terrains these seemingly *more* stable methods actually perform much *worse* in terms of success rates.[3]

Based on our observations, we feel this performance is due to the fact that in challenging terrains, failures are often caused by collision between the robot's legs and the terrain. Moving the COG to a greater extent, while potentially increasing stability, also increases the likelihood of collision with the terrain. Given the fact that our controller is able to maintain stability of the robot with the minimum-length COG trajectories that we plan, we feel that moving the COG more to achieve "greater" stability does more harm than good.

[2]Statistically, over all four terrains the full controller outperforms the controller with no recovery, with no stabilization, with no closed-loop foot placement, and with none of these elements in terms of success probability with p-values of $p = 2.2 \times 10^{-13}$, $p = 0.0078$, $p = 0.0012$, and $p = 5.8 \times 10^{-11}$ respectively, via a pairwise Bernoulli test.

[3]Statistically, our method performs better in terms of success/failure rates than the fixed COG and centered COG methods with p-values of $p = 4.8 \times 10^{-7}$ and $p = 4.9 \times 10^{-9}$ respectively using a pairwise Bernoulli test. Our method performs better in terms of speed with p-values $p = 2.1 \times 10^{-26}$ and $p = 4.6 \times 10^{-34}$ using a pairwise $t$-test.

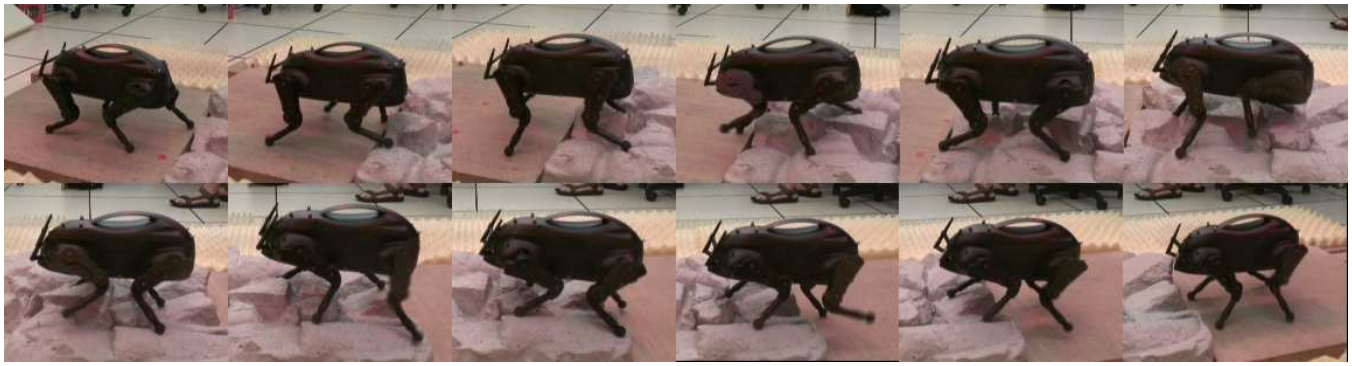


Fig. 7. Snapshots of the robot crossing Terrain #3.

## V. Conclusion

In this paper we presented a hierarchical control system for the LittleDog robot that enables it to navigate over rough terrain. We conducted experimental evaluations which showed that using our control system the robot is able to robustly cross a wide variety of challenging terrains. We also highlighted several elements of the controller that we found to be particularly crucial for robust locomotion, and which are applicable to quadruped robots in general. For these elements we conducted extensive experiments that demonstrated their usefulness in our control system.

## VI. Acknowledgments

This work was supported by the DARPA Learning Locomotion program under contract number FA8650-05-C-7261. This work benefited from numerous discussions with researchers working on the LittleDog robot at Carnegie Mellon, IHMC, MIT, University of Pennsylvania, and USC, and from frequent interaction with the government team running the Learning Locomotion program. We also thank the anonymous reviewers for many helpful suggestions, and for noticing an earlier error in the body stabilization equations.

## References

[1] R. B. McGhee and A. A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331–351, 1968.

[2] V. Hugel and P. Blazevic, "Towards efficient implementation of quadruped gaits with duty factor of 0.75," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

[3] F. Hardarson, "Stability analysis and synthesis of statically balanced walking for quadruped robots," *Royal Institute of Technology*, 2002.

[4] S. Ma, T. Takashi, and H. Waka, "Omnidirectional staic walking of a quadruped robot," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 152–161, 2005.

[5] S. Bai, K. Low, G. Seet, and T. Zielinska, "A new free gait generation for quadrupeds based on primary/secondary gait," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

[6] J. Estremera and P. G. de Santos, "Free gaits for quadruped robots over irregular terrain," *The International Journal of Robotics Research*, vol. 21, no. 2, pp. 115–130, 2005.

[7] ——, "Generating continuous free crab gaits for quadruped robots on irregular terrain," in *Proceedings of the IEEE Transactions on Robotics*, 2005.

[8] H. Lee, Y. Shen, C.-H. Yu, G. Singh, and A. Y. Ng, "Quadruped robot obstacle negotiation via reinforcement learning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006.

[9] M. H. Raibert, *Legged Robots that Balance*. MIT Press, 1986.

[10] J. G. Nichol, S. P. Singh, K. J. Waldron, L. R. P. III, and D. E. Orin, "System design of a quadrupedal galloping machine," *International Journal of Robotics Research*, vol. 23, no. 10–11, pp. 1013–1027, 2004.

[11] I. Poulakakis, J. A. Smith, and M. Buehler, "Modeling and experiments of untethered quadrupedal running with a bounding gait: The scout II robot," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 239–256, 2005.

[12] K. D. Mombaur, R. W. Longman, H. G. Bock, and J. P. Schloder, "Open-loop stable running," *Robotica*, vol. 23, pp. 12–33, 2005.

[13] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *The International Journal of Robotics Research*, vol. 22, pp. 187–202, 2003.

[14] S. Peng, C. P. Lam, and G. R. Cole, "A biologically inspired four legged walking robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[15] H. Kimura, Y. Fukuoka, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 475–490, 2007.

[16] Z. G. Zhang, Y. Fukuoka, and H. Kimura, "Adaptive running of a quadruped robot on irregular terrain based on biological concepts," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[17] D. Pongas, M. Mistry, and S. Schaal, "A robust quadruped walking gait for traversing rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[18] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the littledog quadruped walking on rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[19] J. Z. Kolter, P. Abbeel, and A. Y. Ng, "Hierarchical apprenticeship learning, with application to quadruped locomotion," in *Neural Information Processing Systems 20*, 2007.

[20] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," in *International Conference on Humanoid Robotics*, 2003.

[21] K. Yoneda and S. Hirose, "Dynamic and static fusion gait of a quadruped walking vehicle on a winding path," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992.