2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Fast, Robust Quadruped Locomotion over Challenging Terrain

Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal

Abstract— **We present a control architecture for fast quadruped locomotion over rough terrain. We approach the problem by decomposing it into many sub-systems, in which we apply state-of-the-art learning, planning, optimization and control techniques to achieve robust, fast locomotion. Unique features of our control strategy include: (1) a system that learns optimal foothold choices from expert demonstration using *terrain templates*, (2) a body trajectory optimizer based on the Zero-Moment Point (ZMP) stability criterion, and (3) a floating-base inverse dynamics controller that, in conjunction with force control, allows for robust, compliant locomotion over unperceived obstacles. We evaluate the performance of our controller by testing it on the LittleDog quadruped robot, over a wide variety of rough terrain of varying difficulty levels. We demonstrate the generalization ability of this controller by presenting test results from an independent external test team on terrains that have never been shown to us.**

I. Introduction

Traversing rough terrain with carefully controlled foot placement and the ability to clear major obstacles is what makes legged locomotion such an appealing, and, at least in biology, a highly successful concept. Surprisingly, when reviewing the extensive history of research on robotic legged locomotion, relatively few projects can be found that actually address walking over rough terrain. Most legged robots walk only over flat or at best slightly uneven terrain, a domain where wheeled systems are usually superior.

Recently, several teams working with the LittleDog quadruped robot have been tackling rough terrain, where the obstacles are comparable in size to the leg length. Complete control architectures have been presented that perform locomotion over challenging terrain [1], [2]. Foot placement being one of the most critical aspects of rough terrain locomotion, systems have been proposed that learn optimal foot placement strategies from expert demonstration [3], [4]. Control techniques that enable compliant walking and disturbance rejection have been demonstrated on rough terrain [5].

In this work, we present a complete control architecture for quadruped locomotion over very rough terrain. The controller presented pushes locomotion performance well above that of previously published results. This performance is achieved by the introduction of several novel sub-systems. Optimal foothold selection is learnt from expert demonstration using *terrain templates*, which does away with the need for terrain feature engineering. Body trajectories are optimized for smoothness subject to stability constraints, which allows for very fast, stable locomotion over rough terrain. Finally, a novel floating-base inverse dynamics control law, coupled with force control at the feet enables compliant locomotion that has the ability to deal with unperceived obstacles. Evaluations of our controller are carried out on the LittleDog robot, over terrains of various difficulty levels. Additional testing on our software is carried out by an independent external test team, on terrains that we have never seen, to validate the generalization abilities of the controller.

M.K., J.B., P.P. and S.S. are at the Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089 {kalakris,buchli,pastorsa,sschaal}@usc.edu
M.M. is with Disney Research, Pittsburgh, PA 15213 mmistry@disneyresearch.com

Fig. 1. (a) The LittleDog quadruped robot, manufactured by Boston Dynamics. (b) The pose finder optimizes the pose of the robot body to maximize kinematic reachability while avoiding collisions with the terrain.

The rest of this paper is laid out as follows. In Section II, we describe our experimental setup to provide context for the development of our controller. In Section III, we discuss each component of our controller in detail, highlighting its key and novel features. In Section IV, we present evaluations of our controller on the LittleDog quadruped robot, on various kinds of terrain. Finally, we conclude the paper and discuss ideas for future work in Section V.

II. Experimental Setup

Our experimental setup consists of the LittleDog quadruped robot (Fig. 1(a)), manufactured by Boston Dynamics, with ball-like feet that are close to point feet. The robot is about 0.3 m long, 0.18 m wide, and 0.26 m tall and weighs approximately 2.5 kg. Each leg has 3 degrees of freedom, actuated by geared electric motors. The robot has a *usable* leg length of roughly 13 cm, measured as the perpendicular distance from the bottom of the body to the tip of the foot with the leg in its maximally stretched configuration. LittleDog has a 3-axis force sensor on each foot, position sensors to measure joint angles, and an on-board inertial measurement unit (IMU). An external motion capture system (VICON) provides information about the absolute world position and orientation of the robot.

LittleDog's 12 degrees of freedom are controlled by an on-board computer using PD control at 400 Hz. The rest of our control software runs on an external 4-core Intel Xeon CPU running at 3 GHz, which sends control commands to LittleDog over a wireless connection at 100 Hz.

In order to easily test the performance of our controller on different kinds of terrain, we use a set of interchangeable terrain modules of size 61×61 cm. The terrain modules include flat modules, steps of various step heights, different sizes of barriers, slopes, logs, and rocky terrain of varying difficulty levels. Each terrain module is scanned by a laser scanning

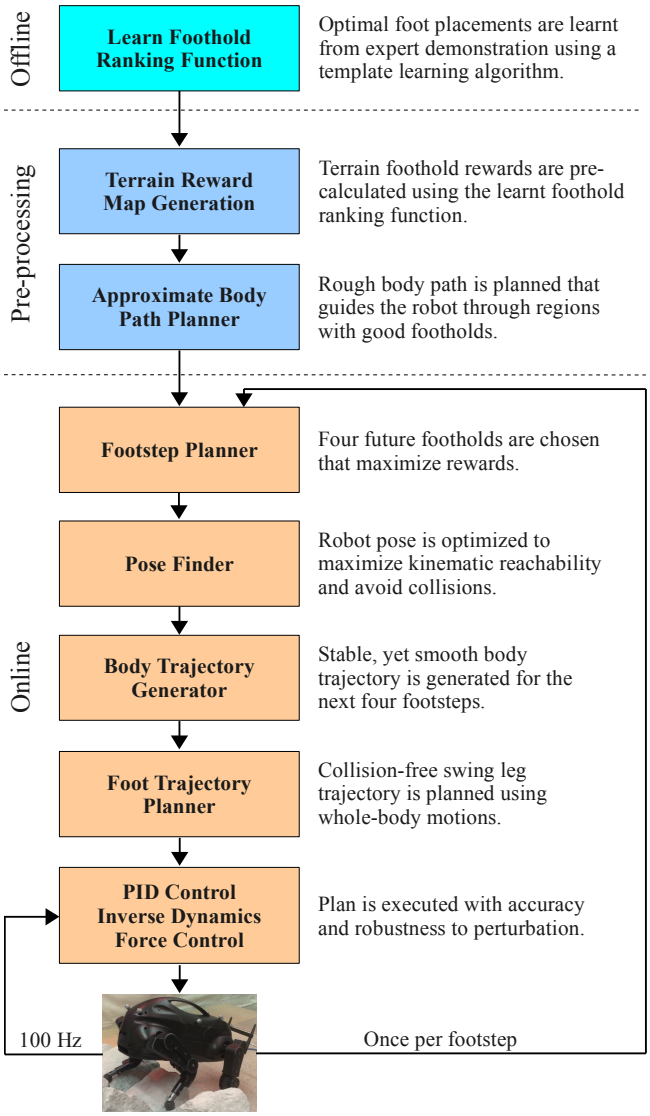


Fig. 2. An overview of our control architecture for quadruped locomotion over rough terrain.

system beforehand, to produce a 1 mm resolution 3D model. The position and orientation of terrain modules are tracked using the motion capture system, This greatly simplifies the perceptual component of rough terrain locomotion, allowing us to focus on the planning and control problems.

An identical setup is available to the five other teams participating in the Learning Locomotion program. This allows for meaningful comparisons between the different control strategies used by the teams.

III. Control Architecture

Finding an optimal sequence of motor commands for 12 degrees of freedom that take the robot from the start to the goal state over rough terrain is considered an intractable problem. Fortunately, such problems tend to have a natural hierarchical decomposition, which we exploit in order to obtain tractable solutions within practical time limits. We perform a limited amount of pre-processing on the terrain before a run, namely, calculation of terrain rewards, and planning of a rough body path through the terrain. All the remaining planning and control is executed online while the robot is walking. This online approach makes our system applicable to domains where terrain information is not available before hand. It also allows for quick re-planning in case of slippage or imprecise execution which is inevitable during fast locomotion on rough terrain. An overview of our control architecture is presented in Fig. 2.

In the following subsections, we describe in detail each sub-system of our approach to quadruped locomotion over rough terrain:

A. Terrain Reward Map Generation

In rough terrain, foothold selection becomes one of the most crucial aspects of robotic legged locomotion. Optimal footholds must be chosen such that the danger of slipping and unplanned collisions are minimized, while stability and forward progress are maximized. However, engineering such a foothold selection function, that scores available footholds based on the terrain features and the state of the robot, turns out to be very difficult. Moreover, finding the set of parameters that perform well on many different kinds of terrain is intractable, due to the number of trade-offs that need to be considered [3].

Instead, we propose an approach that learns a set of terrain features and a foothold ranking function from *good* example foothold choices demonstrated by an expert. The algorithm extracts discretized height maps of small patches of terrain of different scales, so called *terrain templates* [4]. Small terrain templates capture the quality of the actual foothold to avoid slipping, while large terrain templates capture information about collision clearance from terrain and distance to drop-offs. A large set of templates is sampled from a variety of terrain modules. Each template is used in a corresponding feature that measures similarity to the template.

The *template learning algorithm* learns a foothold ranking function by converting the ranking problem into one that can be solved by an off-the-shelf classifier. Feature selection is simultaneously achieved by choosing a sparse, L_1-regularized logistic regression classifier, which learns a classification function that uses a small subset of the available features. The size of the template library is effectively reduced, enabling the footstep planner described in Section III-C to evaluate the foothold ranking function in real-time. Most importantly, the use of terrain templates does away with the need for careful engineering of terrain features, since the appropriate features in the form of templates are automatically selected by the learning algorithm.

Training, collection of expert demonstrations, and learning of the foothold ranking function are performed off-line, while generation of the terrain reward map is done in the pre-processing phase, just before each run. However, since our foothold ranking function also takes the current state of the robot into account to calculate collision scores and other pose-dependent features, the final foothold rewards are computed online. For more information about the learning process, additional terrain and pose features, and evaluations of the generalization performance, please refer to [4].

B. Approximate Body Path Planner

After generating the terrain reward map, we plan an approximate path for the robot body through the terrain,

from the start to the goal. This path guides the robot through regions of the terrain that contain a better choice of footholds.

We discretize the terrain height map into 1 cm square grid cells, and assign a reward to each cell. These rewards are obtained by assuming that the center of the robot is in the cell and the four feet are in their default stance positions. For each leg, the best n (we use $n = 5$) foothold rewards in a small search radius around the default position are summed up to form the final reward for the cell. Using the footholds with the highest rewards around each leg reflects the intentions of the footstep planner, which is to maximize the same rewards.

A policy in this grid is obtained using Dijkstra's algorithm. This provides the optimal path to the goal from every cell in the grid. This approximate body path is then used to direct the lower level footstep planner.

Generation of terrain rewards and planning of the approximate body path are the only pre-processing steps that are required. All the other modules described below are run online while the robot is walking, i.e. planning for the next footstep occurs while the current footstep is being executed. This allows for quick recovery and replanning in case of imprecise planning or execution, and results in a very robust locomotion system.

C. Footstep Planner

Given the cached terrain reward map and the current state of the robot, the footstep planner calculates optimal foothold choices for the next four steps. Optimality is based on the foothold reward function which was learnt from expert demonstration, as described in Section III-A. These rewards are biased to guide the robot towards the approximate body path plan. Search complexity is reduced by following a fixed sequence of leg movements (left hind, left front, right hind, right front), that has been shown to be the optimum sequence that maximizes static stability [6].

Our planner computes the next four optimal foothold choices, using a greedy search procedure at each step. For every candidate foothold choice, we optimize the pose of the robot body using an approximate *pose finder*, described further in Section III-D. The final reward for each candidate foothold is a combination of the cached terrain reward and additional features based on the robot pose, such as in-radius of future support triangles and collision cost features.

In some situations, such as crossing a large gap, the use of a greedy search procedure is inadequate. Choosing a footstep in a greedy fashion can put the robot in a configuration where it cannot proceed, or has to make a bad foothold choice to do so. When faced with such terrain, we use the Anytime Repairing A* (ARA*) algorithm [7] to find a plan that maximizes the sum of foothold rewards from the start to the goal. ARA* is an *anytime* variant of the heuristic search algorithm A*. It starts with a sub-optimal plan and then iteratively improves the plan until terminated. The anytime property of ARA* allows us to terminate it after a fixed amount of time, and use the current best plan found by the algorithm. Since this planner needs to be executed in the pre-processing phase, i.e. before execution, we use this planner only on terrains where the greedy planner is known to perform poorly, currently only large gaps that are as long as the legs of the robot (e.g., Fig 1(b))). For all other classes of terrain, the advantage of running ARA* before execution was found to be small, hence we prefer to minimize pre-processing time by using the online, greedy planner.

D. Pose Finder

Given the 3-D locations of the stance feet, the pose finder optimizes the 6-D pose of the robot body in order to maximize kinematic reachability and to avoid configurations that collide with the terrain. The x and y positions of the body are set to the in-center of the support triangle, to be modified later by the body trajectory generator (Sec. III-E). This leaves the z position of the body and its roll, pitch and yaw angles as the search variables for body pose optimization. The kinematics of the LittleDog robot are such that the 3-D positions of all four feet and the 6-D pose of the robot body are sufficient to fully specify the 12 joint angles; i.e, there are no extra redundancies to be optimized.

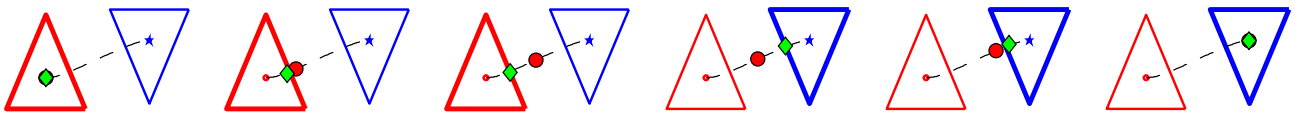
We use two different versions of the pose finder: the first is a very fast, approximate search over a coarse grid of the free variables. This coarse search is used to optimize the pose of every candidate foothold evaluated by the foothold planner. The second version is a more exact, gradient-based optimizer. We use a floating-base iterative inverse kinematics algorithm [8] to satisfy the locations of the feet, while optimizing the secondary objectives of collision avoidance and maintenance of a default posture in the null space of the first. This fine-grained optimization is only performed for the final foothold selected by the footstep planner.

E. Body Trajectory Generator

The body trajectory generator uses the next four planned footholds and creates a body trajectory. The generated trajectory is smooth, stable with respect to the Zero-Moment Point (ZMP), a dynamic stability criterion, and works even over highly irregular foot placement patterns, which is almost always the case in rough terrain. We also eliminate the *four leg support phase*, which is time spent moving the body while all four feet remain on the ground. This phase is required by traditional body trajectory planners that use the Center of Gravity (COG) as a stability criterion. The idea of these planners is to keep the COG over the triangle formed by the stance feet, called the *support triangle*, while a leg is swinging. These support triangles are typically shrunk by a stability margin to account for inaccurate execution. A four leg support phase is required when two consecutive shrunk support triangles are disjoint. Keeping the duration of the four leg support phase low, or eliminating it completely results in increased locomotion speed. COG-based planners are further limited in their locomotion speed by the fact that the COG is a purely static stability criterion; it neglects the effects of acceleration on the stability of the robot.

We instead use the ZMP [9] as our stability criterion. We represent the trajectory of the COG as a series of quintic spline segments (fifth order polynomials) and formulate body trajectory generation as an optimization problem to minimize squared accelerations along the trajectory, subject to continuity and ZMP stability constraints. The resulting optimization is a convex quadratic program (QP) that can be solved efficiently by off-the-shelf solvers.

Swing leg durations are heuristically decided based on the expected travel distance of the foot. The body trajectory



(a) $t = 0.0$ s (b) $t = 0.4$ s (c) $t = 0.5$ s (d) $t = 0.51$ s (e) $t = 0.6$ s (f) $t = 1.0$ s

Fig. 3. Snapshots from a ZMP-stable body trajectory generated by our optimizer. The COG (red circle) moves smoothly between the two disjoint support triangles, while the ZMP (green diamond) is always kept within the *current* support triangle (marked in bold in each frame). At the moment of support triangle switching (in between $t = 0.5$ s and $t = 0.51$ s), the ZMP position discretely switches between them, achieved by manipulation of the acceleration profile of the COG, which influences the ZMP position according to Eqn. 1.

during each swing leg movement is split up into n quintic spline segments (we use $n = 3$). The cost function is defined as the integral of squared acceleration along the trajectory; this expression is quadratic in the spline coefficients.

Double differentiability of the body trajectory is important when using inverse dynamics control, to guarantee continuous motor torques. Each individual spline segment, being a fifth order polynomial, is twice differentiable. To guarantee this property in between spline segments, constraints are constructed by equating the position, velocity and acceleration at the junction of two neighboring spline segments. These constraints are linear in the spline coefficients. Constraints are also added to ensure that the start of the trajectory matches the current position, velocity and acceleration.

In order to formulate stability constraints, we consider the cart-table ZMP model [10] as an approximation of the LittleDog robot, which has a heavy body relative to its legs. This model relates the COG and ZMP positions as follows:

$$x_{zmp} = x_m - \frac{z_m \ddot{x}_m}{\ddot{z}_m + g}, \tag{1}$$

where x_{zmp} is the position of the ZMP along the x axis, x_m and z_m are the positions of the COG along the x and z axes respectively, g is the acceleration due to gravity, and accelerations of the COG along the x and z axes are denoted by $\ddot{x}_m$ and $\ddot{z}_m$. The position of the ZMP along the y axis is obtained by replacing all occurrences of x with y in Eqn 1.

At every instant of time, the ZMP must lie in the support triangle, or, more generally, the convex hull of the support polygon. This can be written mathematically as a set of k constraints, k being the number of sides in the convex hull of the support polygon. For each line segment connecting two consecutive points on the convex hull, the ZMP must lie on the interior side of it. To represent this constraint for all times, we discretize the trajectory at the control interval (10ms), and express the ZMP constraints at each of these times. These constraints are linear in the spline coefficients.

The optimization problem thus formulated is convex, with a quadratic objective function and linear constraints. This can be readily and efficiently solved using off-the-shelf QP solvers. We use the freely available QuadProg++ library [11] to perform the optimization, which is typically completed in 30ms. Situations may arise where the optimization fails, i.e. there is no feasible trajectory that can satisfy all the constraints. In such a case, we successively relax the problem, by reducing stability margins and increasing four leg support phase durations, until the optimization succeeds.

The body trajectory generator described above optimizes the trajectory over the four future footholds selected by the foothold planner. Running the optimization in this *receding horizon* manner, combined with suitable stability margins provides us with smooth and stable body trajectories, even over very rough terrain, without special parameter tuning for different terrain classes. Under this framework, we can drop the requirement for a four-leg support phase altogether, even when moving between two disjoint support triangles. This is possible by manipulation of the acceleration profile of the trajectory such that the ZMP jumps instantaneously from one support triangle to the next. As shown in Fig. 3, the optimizer automatically finds such solutions. However, this requires us to drop the constraint on continuous accelerations at the instant when we switch between support triangles. In practice, we insert a tiny (50ms) four leg support phase, which is considerably smaller than that required by traditional COG-based trajectory planners. This provides us with continuous accelerations which are necessary for our inverse dynamics controller, as described in Section III-H.

F. Foot Trajectory Planner

Given a stable body trajectory (provided by the body trajectory generator) and the desired footstep locations (provided by the footstep planner), the foot trajectory planner generates a trajectory for the swing leg that avoids collisions with the terrain. An initial trajectory is generated that guides the end-effector over the terrain from its initial position to its goal. This trajectory is subsequently optimized to eliminate potential shin or knee collisions.

An initial trajectory is generated that moves the foot over the terrain in the plane containing the start and the goal. Points on the convex hull of the terrain along the plane from the start to the goal are raised by a clearance height. A trajectory is then generated through these points using piece-wise quintic splines optimized for minimum acceleration. This trajectory is sampled at 10ms intervals and converted into joint angles using the standard analytical inverse kinematics solution for a 3-DOF arm. Kinematic infeasibilities are simply ignored, since they are taken care of in the subsequent optimization phase.

The initial trajectory is only guaranteed to avoid collisions of the end-effector with the terrain, however, it may still contain knee or shin collisions. Moreover, if parts of the trajectory were kinematically infeasible, they may be in collision with the end-effector as well. We run a trajectory optimizer called CHOMP (Covariant Hamiltonian Optimization for Motion Planning) [12] on the trajectory to eliminate any collisions that remain. CHOMP also serves to smooth the trajectory in joint space according to a quadratic cost function; we choose a minimum squared jerk cost function to provide continuous accelerations for inverse dynamics control. CHOMP uses covariant gradient descent in order to push the trajectory away from obstacles while still maintaining its smoothness. The cost function for collisions (and

its gradients) is defined based on a signed distance field (SDF), which is a 3-D grid of distances to the closest obstacle boundary. The sign of the SDF voxel values are positive if it is outside obstacles, and negative otherwise. The SDF is generated before the start of the trial as a pre-processing step. Joint limit violations are handled using a covariant update that eliminates them without impacting smoothness. We run CHOMP on a virtual 8-DOF system: the 3 joints of the leg, the body z height, and 4 variables that together represent the body quaternion. This enables the use of whole-body motions such as pitching in order to avoid swing leg collisions.

In our system, footstep planning and trajectory generation is performed while the robot is executing a swing motion with one of its legs. Hence, it is critical that the entire trajectory for the next swing leg be generated before the current swing motion ends. Our implementation of CHOMP is most often able to produce collision-free, smooth trajectories in less than 100 iterations, which typically takes no more than 50ms. The lower limit on swing leg duration in our system is 300ms, allowing us to run the optimizer while the robot is walking, one step in advance, as opposed to pre-generating trajectories all the way to the goal before execution.

G. Dynamic Motions for Extreme Terrain

Terrain that contain steps and barriers of height up to 12 cm and gaps of width up to 17 cm push the LittleDog robot beyond its limits of kinematic feasibility. Since the maximum ground clearance the robot can achieve with its legs fully stretched is roughly 13 cm, it is not possible to safely walk over such obstacles. Therefore a special series of movements, in particular a lunging movement and a sliding movement, were designed to enable the robot to cross such obstacles. Similar dynamic motions have been demonstrated before on the LittleDog robot [13].

The dynamic lunging motion involves shifting the weight of the robot onto its hind feet, pitching up its nose, and lifting its feet onto the step or barrier. In case of lunging across a gap the robot executes a bounding-like movement before the lunge to ensure that the COG reaches the other side of the gap (using its momentum). The sliding movement involves shifting the robot's weight in front of its front legs and using the robot's nose as a fifth stance to lift its rear, which allows for collision free hind leg movements. To account for different COG positions and different friction properties across different LittleDog robots, a calibration procedure has been developed that performs a series of lunging movements to estimate the optimal parameter set for that particular robot.

H. Execution and Control

The 6-D pose trajectory of the body is converted into joint angle trajectories for the 3 stance legs using a closed form inverse kinematics solution for each 3-DOF leg individually. These trajectories are then combined with the swing leg joint trajectories to form the final sequence of joint angles to be executed on the robot.

In order to guarantee stability of the robot and accurate foot placement, it is essential that the planned joint trajectories be executed accurately. At the same time, the robot has to be able to overcome small perturbations and slips, and robustly handle unperceived terrain. To achieve this level of robustness, we use a combination of five controllers, as described below:

1) PD control (400 Hz): The desired joint angle positions and velocities are sent over a wireless link to LittleDog's on-board computer that servos these commands using PD control at 400 Hz.
2) Inverse dynamics (100 Hz): We use a novel, analytical floating-base inverse dynamics algorithm [14] that does not require knowledge of the contact forces at the feet. The joint torques thus computed are sent to the LittleDog at 100 Hz, which it then applies as feed forward torques, in addition to the torques from the PD controllers. Using inverse dynamics provides better tracking of trajectories, and allows us to use lower PD gains than otherwise possible, making the robot more compliant to perturbations [5].
3) Force control (400 Hz): The above inverse dynamics algorithm provides us with the expected contact forces at all four feet. This allows us to control the forces at each foot using the force sensor data as feedback. The use of force control allows us to negotiate *unperceived* obstacles up to 4 cm in height. These results are discussed in detail in a previous publication [5].
4) Closed loop body pose control (100 Hz): We monitor the 6-D robot pose as registered by the motion capture system, and use an integral controller to track the desired pose. The controller performs floating base inverse kinematics to servo body position and orientation, in the null space of foot location constraints. This modifies the desired joint angles that are sent to the PD and inverse dynamics controllers. Tracking the body pose keeps the robot stable in spite of small slips at the stance feet.
5) Closed loop foot position control (100 Hz): We use a similar integral controller on the swing foot position. This controller significantly improves foot placement accuracy, which is critical for walking on rough terrain.

IV. Evaluations

We now present evaluations of our LittleDog control software on the terrain modules that are present in our lab, as well as results from tests conducted by an external test team on terrains that we have never seen.

In our lab, we tested our controller on rocky terrain of various difficulty levels, round rocks/pebbles (not movable), and wavy dunes. The most difficult rocky terrain contains obstacles up to 10 cm (76% of the usable leg length) in height. We also tested our software on geometric test scenarios like gaps, steps of varying step heights, slopes, and barriers. Our controller is able to traverse all classes of terrain at speeds ranging from 7.2 cm/s to 13 cm/s (0.24 to 0.43 body lengths/s). The advantage of our ability to replan quickly in case of a deviation from the plan is evident when testing on more challenging rocky terrain, where the robot succeeds with high probability even after slipping or imprecise foot placement. Steps up to a height of 10 cm can be climbed using our regular walking gait, and lunging can be performed to traverse steps up to 12 cm in height. Similarly, lunging is employed to cross barriers up to 12 cm in height. Gaps up to 15 cm wide can be traversed successfully by generating

Fig. 4. Sequence of snapshots of the quadruped robot LittleDog crossing a test terrain module using the controller presented in this work.

a footstep plan using the ARA* planner, and lunging can again be employed to cross gaps up to 17 cm wide. The choice of high-level strategies (i.e., ARA* planner, lunging) is currently specified by the user per terrain class. Automatic terrain class detection is outside the scope of this work, and may be addressed in future.

Inverse dynamics control, coupled with force control and low PD gains allow us to overcome *unperceived* obstacles. This property was tested by placing wooden planks on the terrain, which was not sensed by the motion capture system. Our controller was able to handle obstacle heights up to 4 cm without significant difficulties. Additionally, a movable see-saw was set up on the terrain for the robot to walk over, once again, unsensed by the motion capture system. This scenario posed no problems for our software, demonstrating the robustness added by this combination of controllers.

Fig. 4 shows the robot traversing one of the test terrain modules. The attached video highlights the key features of our controller, and contains a compilation of video from test runs of the robot on different kinds of terrain.

Our control software was subject to additional testing by an external test team, on terrain modules that have never been shown to us. As of the final (privately disclosed) test results, the LittleDog robot controlled by our software has successfully traversed all the hidden test scenarios posed by them. These results serve as a testament to the true generalization ability of our controller.

V. Conclusion

We have presented a general quadruped locomotion controller for fast locomotion over rough terrain, and its application on the LittleDog robot. This controller relies on a decomposition of the problem into many sub-systems, in which we apply state-of-the-art learning, planning, optimization and control techniques to achieve high performance. Novel features of our controller include a procedure that learns optimal foothold choices using terrain templates, a body trajectory optimizer based on the ZMP stability criterion, and a floating-base inverse dynamics controller that does not require knowledge of the contact forces. Evaluations presented have shown that our controller is highly performant and generalizes well on many different kinds of terrain.

This controller represents our final submission to the Learning Locomotion program, in which five other teams also took part. Our approach has proven to be very competitive. Most of the techniques developed in this work are generally applicable to planning and control problems on other systems. In future work we intend to apply these techniques to other platforms such as humanoid robots.

Acknowledgements

We would like to acknowledge the contributions of Dimitris Pongas, Jan Peters and Jo-Anne Ting to previous versions of our control software. Regular testing carried out by the Learning Locomotion Government Team has been very helpful in the development of this work. This research was supported in part by the DARPA program on Learning Locomotion, National Science Foundation grants ECS-0326095, IIS-0535282,CNS-0619937, IIS-0917318, CBET-0922784, EECS-0926052, the Okawa Foundation, and the ATR Computational Neuroscience Laboratories. J.B. was supported by a prospective researcher fellowship from the Swiss National Science Foundation.

References

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

[2] 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 Int. Conf. on Robotics and Automation*, 2007, pp. 1467–1473.

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

[4] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 167–172.

[5] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 814–820.

[6] R. B. McGhee and A. A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, no. 1-2, pp. 331–351, Aug. 1968.

[7] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: anytime a* with provable bounds on suboptimality," *Advances In Neural Information Processing Systems*, 2003.

[8] M. Mistry, J. Nakanishi, G. Cheng, and S. Schaal, "Inverse kinematics with floating base and constraints for full body humanoid robot control," in *8th IEEE-RAS Int. Conf. on Humanoids*, 2008, pp. 22–27.

[9] M. Vukobratovic and B. Borovac, "Zero-moment point - thirty five years of its life," *Int. J. of Humanoid Robotics*, vol. 1, no. 1, pp. 157–173, 2004.

[10] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, 2003, pp. 1620–1626.

[11] L. D. Gaspero, "Quadprog++, a c++ library for quadratic programming," http://quadprog.sourceforge.net/.

[12] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: gradient optimization techniques for efficient motion planning," in *Proceedings of the 2009 IEEE Int. Conf. on Robotics and Automation*, 2009.

[13] K. Byl, A. Shkolnik, S. Prentice, N. Roy, and R. Tedrake, "Reliable dynamic motions for a stiff quadruped," in *Experimental Robotics*, 2009, pp. 319–328.

[14] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition," in *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, 2010.