The International Journal of
Robotics Research
30(2) 236–258

DOI: 10.1177/0278364910388677
ijr.sagepub.com

Learning, planning, and control for quadruped locomotion over challenging terrain

Mrinal Kalakrishnan[1], Jonas Buchli[1], Peter Pastor[1], Michael Mistry[2] and Stefan Schaal[1]

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 terrains of varying difficulty levels. The terrain that the robot was tested on includes rocks, logs, steps, barriers, and gaps, with obstacle sizes up to the leg length of the robot. We demonstrate the generalization ability of this controller by presenting results from testing performed by an independent external test team on terrain that has never been shown to us.*

Keywords

Quadruped locomotion, locomotion planning and control, template learning, ZMP optimization, floating base inverse dynamics

1. 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 nature, 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.

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 subsystems. Optimal foothold selection is learnt from expert demonstration using *terrain templates*, which does away with the need for careful engineering of features that quantify the suitability of a foothold for locomotion. 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 terrain of various difficulty levels. Additional testing on our software is carried out by an independent external test team, on terrain that we have never seen, to validate the generalization abilities of the controller.

The rest of this paper is laid out as follows. Related work on robotic legged locomotion is discussed in Section 2. In Section 3, we describe the Learning Locomotion program, experimental setup, test procedures, and evaluation metrics. In Section 4, we discuss the structure of our control architecture, and briefly describe every component of it. Sections 5, 6, and 7 contain detailed descriptions and results from the key components of our locomotion system: foothold learning using terrain templates, body trajectory optimization using Zero-Moment Point (ZMP) constraints, and robust locomotion using inverse dynamics and force

[1]Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089, USA
[2]Disney Research, Pittsburgh, PA 15213, USA

Corresponding author:
Mrinal Kalakrishnan
Computational Learning and Motor Control Lab
University of Southern California, Los Angeles, CA 90089, USA.
Email: kalakris@usc.edu

Table 1. Number of templates learnt on each spatial scale, from a total of 946, with and without augmentation by features

	Small scale	Medium scale	Large scale
Templates + features	1	19	23
Templates only	10	24	28

control. In Section 8, we present evaluations of our controller on the LittleDog quadruped robot, on various kinds of terrain. We present a discussion of certain points of interest in Section 9. Finally, we conclude the paper and discuss ideas for future work in Section 10.

Fig. 1. The LittleDog quadruped robot on rocky terrain.

2. Related Work

Early research on robotic legged locomotion focused on gaits that are statically stable, i.e. the center of gravity (COG) of the robot is always kept over the polygon formed by the supporting feet. McGhee built the first computer-controlled quadruped robot, Phony Pony (McGhee 1967), and studied the stability properties of various quadruped gait sequences (McGhee and Frank 1968). Hirose et al. (1985) built a series of quadruped robots called TITAN that could climb stairs using static gaits.

A breakthrough in legged locomotion was achieved by Raibert et al. when he built a series of legged robots that achieved high-speed locomotion with dynamic balancing (Raibert 1986; Hodgins and Raibert 1991). This line of research has recently resulted in the BigDog robot (Raibert et al. 2008) built by Boston Dynamics, which is a quadruped that uses a dynamically balanced trot gait to traverse rough terrain.

Another important wave of research was created by biologically inspired robot designs (Saranli et al. 2001; Fukuoka et al. 2003; Cham et al. 2004; Smith and Poulakakis 2004), that try to overcome the limited ability of multi-legged robots in rough terrain with biomimetic hardware design principles. In the realm of biologically inspired legged locomotion, there has also been research on biologically inspired controllers, usually in the form of central pattern generators (CPGs) (Ijspeert 2008). The CPG approach can be seen as the control-based analog to biomimetic robot design, i.e. in both approaches, robustness of the controller is sought in the emergent attractor dynamics between the robot, controller, and environment.

Machine learning for locomotion has been used to acquire quadruped locomotion gait patterns (Murao et al. 2001) and optimize them for performance (Kohl and Stone 2004), primarily over flat ground. Recently, the DARPA Learning Locomotion program spawned off a line of research on applying machine learning and optimization techniques to the problem of legged locomotion over rough terrain, where the obstacles are comparable in size to the leg length of the robot. Complete control architectures have been presented that perform locomotion over challenging terrain (Rebula et al. 2007; Kolter et al. 2008; Zucker et al. 2010; Kalakrishnan et al. 2010). 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 (Kolter et al. 2007; Kalakrishnan et al. 2009). Control techniques that enable compliant walking and disturbance rejection have been demonstrated on rough terrain (Buchli et al. 2009). Our work has also been developed in the context of the DARPA Learning Locomotion project, which we introduce in the next section.

3. Background

In the following sections, we provide some background information and context to the technical work conducted. We discuss the DARPA Learning Locomotion program, and describe the standardized experimental setup, test procedures and metrics.

3.1. Learning Locomotion

The DARPA Learning Locomotion program was set up to push the performance of robotic legged locomotion, both in terms of speed and robustness, over very rough terrain. Six teams were provided identical hardware and test setups, which allows for meaningful comparisons to be made between the different control strategies used by the teams. Software developed by the teams was subject to monthly testing by an independent test team, on terrain that has never been shown to the teams. The program was split into three phases, with successively increasing metrics for minimum speed and obstacle heights. These metrics and test procedures are detailed in Section 3.3.

3.2. Experimental Setup

Our experimental setup consists of the LittleDog quadruped robot (Figure 1) manufactured by Boston Dynamics, with

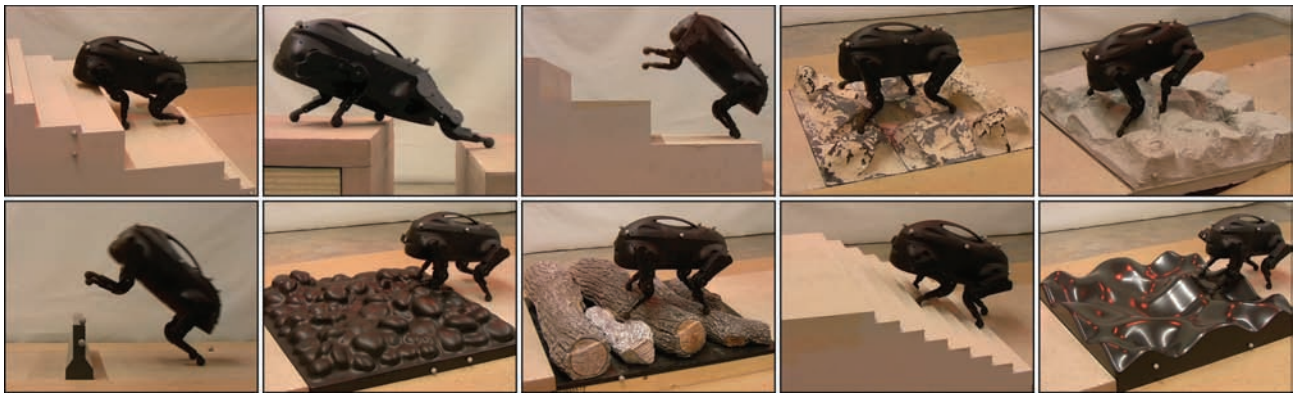

Fig. 2. LittleDog crossing a variety of terrains.

ball-like feet that are close to point feet. It is about 0.3 m long, 0.18 m wide, 0.26 m tall, and weighs approximately 2.5 kg. Each leg has three degrees of freedom (DOFs), 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 three-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 at a rate of 100 Hz. LittleDog's 12 DOFs are controlled by an on-board computer using PD control at 400 Hz. The rest of our control software runs on an external quad-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 cm $\times$ 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. Figure 2 shows a sampling of the terrains that are available to us for testing.

Each terrain module is scanned beforehand by a laser scanning system, to produce a high-resolution (1 mm) 3D model. A unique arrangement of reflective markers is fixed on each terrain module, enabling the motion capture system to track the position and orientation of each module independently, thus providing complete information about the terrain in the world. This greatly simplifies the perceptual component of rough terrain locomotion, allowing us to focus on the planning and control problems. The terrain modules are not moved during each trial, i.e. one run of the robot from the start to the goal location.

3.3. Testing and Evaluation Metrics

In the final tests of Phase 3 of the Learning Locomotion program, teams were required to submit LittleDog control code to traverse seven different classes of terrains. These terrains had never been seen by the teams before; only the terrain classes were known. Each terrain had to be crossed at a minimum speed of 7.2 cm s^{-1} (0.24 body lengths per second) and contained obstacles of up to 10.8 cm (83% of the usable leg length) in height. Three trials were conducted over each terrain, of which the best two were considered for evaluation. This test procedure and metric encourages the teams to create controllers that can generalize to novel situations, behave robustly, and achieve a high locomotion speed.

4. Control Architecture

Finding an optimal sequence of motor commands for 12 DOFs 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 each trial, namely, calculation of terrain rewards, and planning of a rough body path through the terrain. All of 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 Figure 3.

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

4.1. 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

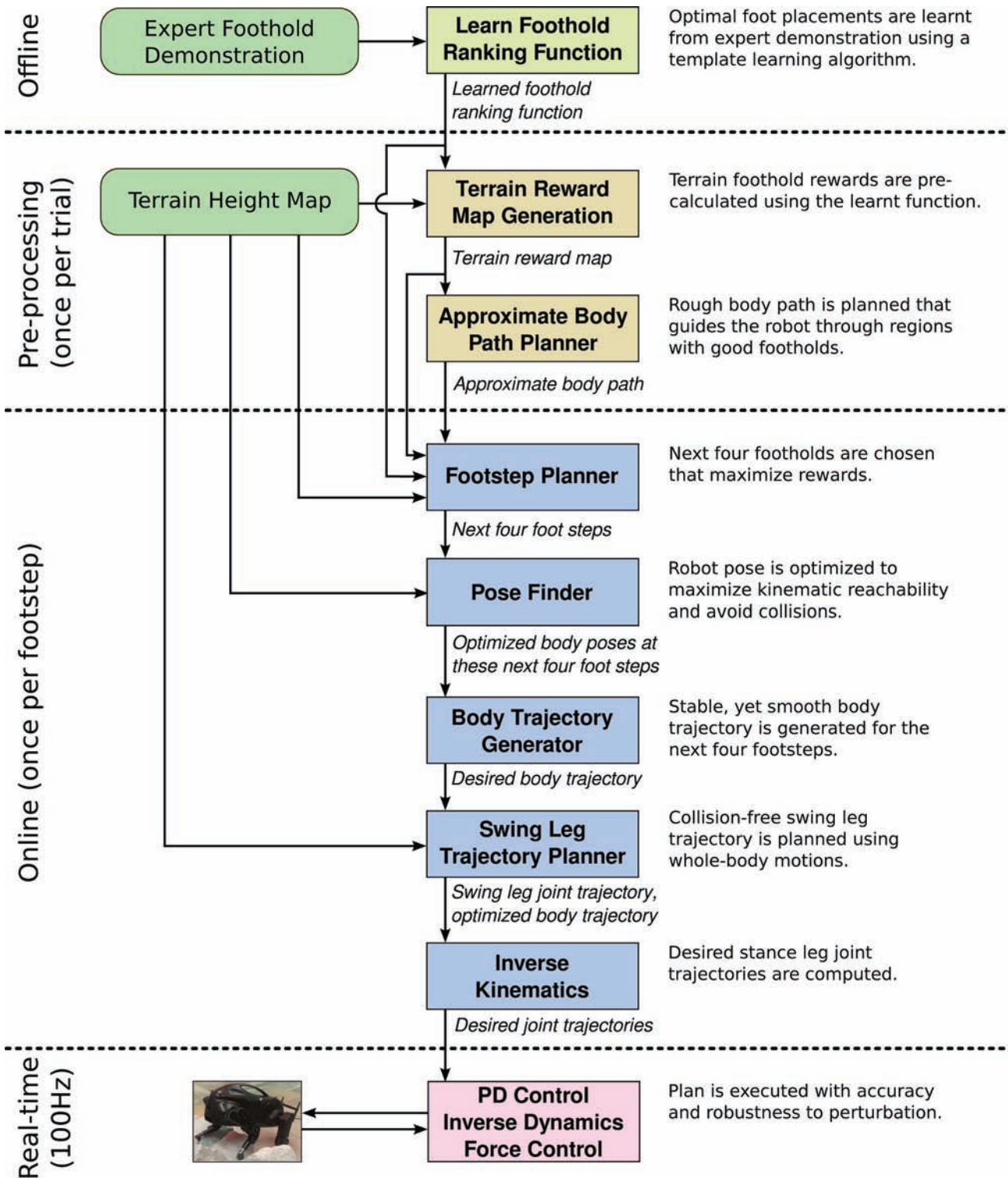

Fig. 3. An overview of our control architecture for quadruped locomotion over rough terrain.

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, owing to the number of trade-offs that need to be considered (Kolter et al. 2007).

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. We

provide only a high-level overview of this approach here, full details can be found in Section 5. The algorithm extracts discretized height maps of small patches of terrain of different scales, so-called *terrain templates* (Kalakrishnan et al. 2009). 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. A suitable distance metric is used to measure the similarity of a candidate foothold with each template, forming a feature vector that is used to rank footholds.

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 4.3 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 trial. 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. More information about the learning process, additional terrain and pose features, and evaluations of the generalization performance are presented in Section 5.

4.2. 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 is intended to guide the robot through regions of the terrain that contain a better choice of footholds.

We discretize the entire terrain into 1 cm square grid cells, and assign a reward to each cell. The cell 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 foothold rewards.

The final path is obtained by generating a policy in this grid using Dijkstra's algorithm (Dijkstra 1959), which is equivalent to solving for the value function at every state using dynamic programming (Sniedovich 2006). 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 two pre-processing steps that are required. All of 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 the case of imprecise planning or execution, and results in a very robust locomotion system.[1]

4.3. 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 4.1. 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 (McGhee and Frank 1968).

Our planner computes optimal foothold choices for the next four steps by 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 4.4. 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.

The greedy search technique turns out to be inadequate in certain scenarios that are kinematically challenging, such as crossing a very large gap. Choosing a footstep in a greedy fashion can put the robot in a configuration from which 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 (Likhachev et al. 2003) 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*, that 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. We run the ARA* planner for 30 seconds in the pre-processing phase, to generate a complete footstep plan from the start to the goal. We use this planner only to traverse terrain that the greedy planner is known to perform poorly on, currently only large gaps that are as long as the legs of the robot (e.g. Figure 4). For all other classes

Fig. 4. The pose finder optimizes the pose of the robot body in order to maximize kinematic reachability while avoiding collisions with the terrain.

of terrain, the advantage of running ARA* before execution was found to be small, hence we prefer to minimize the pre-processing time by using the online, greedy planner. The terrain class is currently specified by the user per terrain module; automatic terrain class detection is outside the scope of this work.

4.4. Pose Finder

Given the 3D locations of the stance feet, the pose finder optimizes the 6D pose of the robot body in order to maximize the kinematic reachability and to avoid configurations that collide with the terrain. The x and y positions of the body are fixed based on stability criteria. 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 3D positions of all four feet and the 6D pose of the robot body are sufficient to fully specify the 12 joint angles, i.e. there are no extra redundancies that need to be optimized.

We use two different versions of the pose finder. In the first, we use the cyclic coordinate descent search technique (Bertsekas et al. 1999) over a very coarse grid of the free variables (body roll, pitch, yaw, and z height). This pose finder returns an approximate solution very fast, and 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 (Mistry et al. 2008) 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 provides a locally optimal solution at the expense of higher computation time when compared with the discrete search. This optimization is only performed on the sequence of footholds selected by the footstep planner.

4.5. 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. In contrast to traditional COG-based trajectory planners, our body trajectory generator eliminates the four-leg support phase, which is the duration of time spent moving the body with all four feet on the ground. This ensures that one leg is swinging at all times, resulting in increased locomotion speed, without sacrificing stability.

We formulate the body trajectory generation problem as an optimization problem. The COG trajectory is represented as a series of quintic spline segments, and the cost function is to minimize squared accelerations along the trajectory. This function is quadratic in the spline coefficients. We use the ZMP (Vukobratovic and Borovac 2004) as our stability criterion. To attain stability, the ZMP must lie within the polygon formed by the supporting feet, at all times. At each time step, these constraints can be represented as linear functions of the spline coefficients. The resulting optimization is a convex quadratic program (QP) that can be solved efficiently by off-the-shelf solvers. This optimization provides a smooth and stable body trajectory for locomotion over rough terrain. Details about this formulation of body trajectory optimization as a convex optimization problem are presented in Section 6.

4.6. 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 vertical 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 piecewise quintic splines optimized for minimum acceleration. This trajectory is sampled at 10-ms 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 initial trajectory were kinematically infeasible, these configurations may contain collisions of the end-effector with

the terrain. We run a trajectory optimizer called CHOMP (Covariant Hamiltonian Optimization for Motion Planning) (Ratliff et al. 2009) 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) are defined based on a signed distance field (SDF), which is a 3D grid of distances to the closest obstacle boundary. The sign of the SDF voxel values are positive if they are 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 three joints of the leg, the body z height, and four 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 50 ms. The lower limit on swing leg duration in our system is 300 ms, allowing us to run the optimizer while the robot is walking, one step in advance, as opposed to pre-generating trajectories all of the way to the goal before execution as in Ratliff et al. (2009). In the rare occasion that 100 iterations of CHOMP do not produce a collision-free trajectory, we optimistically execute the final trajectory as-is, and leave it to the execution module (Section 4.9) to maintain stability, and eventually recover and replan.

4.7. 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 (Byl et al. 2009).

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 the 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.

4.8. Sensor Preprocessing

The LittleDog robot has two sources of orientation measurement, the VICON motion capture system, and the on-board inertial measurement unit (IMU). Both provide a quaternion q of the robot's orientation: q_{MOCAP} from the motion capture system and q_{IMU} from the IMU. Here q_{MOCAP} contains outliers and noise, but provides an absolute orientation measurement, without drift and q_{IMU}, on the other hand, is obtained through integration, which provides a clean signal, but drifts slowly over time due to integration errors. The performance of control algorithms can be improved if the two sources of orientation can be combined to compensate for the shortcomings of one another. We developed a Bayesian outlier detection and removal algorithm, which estimates the offset between q_{IMU} and q_{MOCAP}, which is a slowly changing signal infested with noise and outliers. This Bayesian regression algorithm operates incrementally, in real time, providing us with cleaner orientation and angular velocity estimates than either source provides on its own. Details and results from this algorithm can be found in a previous publication (Ting et al. 2007).

4.9. Execution and Control

The 6D pose trajectory of the body is converted into joint angle trajectories for the three 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 in the following. A graphical overview of these controllers is shown in Figure 5.

1. Joint 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 (Mistry

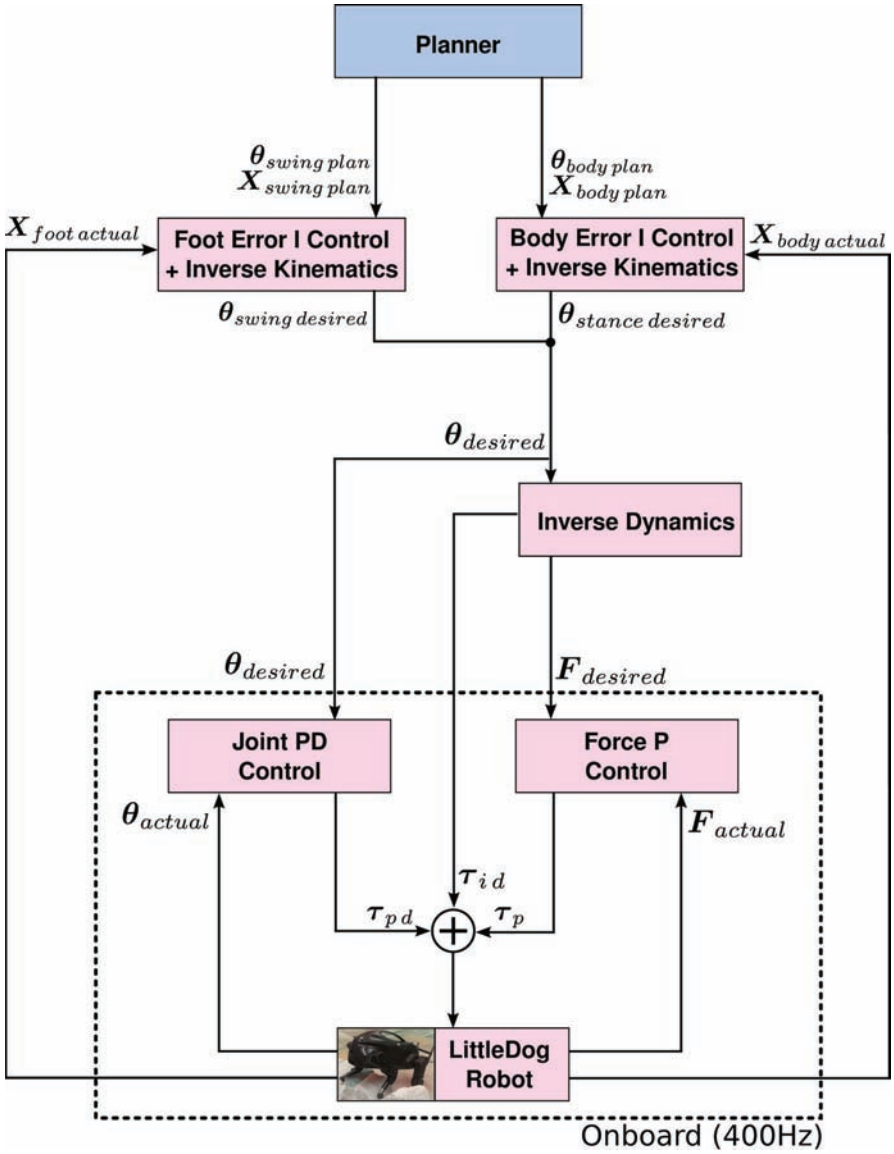

Fig. 5. Flow diagram of our low-level control software. The joint PD and force P controllers run on the robot at 400 Hz, while the remaining controllers operate on the off-board computer and send commands to the robot at 100 Hz.

2009) 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 (Buchli et al. 2009). This controller is discussed in detail in Section 7.

3. Force P 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 in conjunction with inverse dynamics allows us to negotiate *unperceived* obstacles up to 4 cm in height. These results are discussed in detail in Section 7.
4. Body error integral control (100 Hz): We monitor the 6D robot pose as registered by the motion capture system, and use an integral (I) 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. Foot error integral 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.

5. Learning a Ranking Function for Foothold Selection using Terrain Templates

In this section, we describe our approach to learning an optimal foothold selection policy from expert demonstration. We develop this work in the context of quadruped locomotion with the LittleDog robot, but this approach in principle scales to any other legged locomotion system. We consider static or quasi-static locomotion, in which either the COG or the ZMP (Vukobratovic and Borovac 2004), respectively, is always maintained within the support polygon. The task is to find a suitable foothold for the swing leg, given the positions of the stance legs. The chosen foothold must:

- minimize slipping on terrain;
- not result in collisions of the robot with the environment;
- maximize forward progress towards the goal;
- be within the kinematic range of the robot;
- serve to maximize the area of future support polygons for maximum stability.

Most often, these goals are conflicting, and trade-offs must be made between them. Specifying these trade-offs by hand is non-trivial, since it typically involves manual tuning of a large number of parameters. From our experience, it is nearly impossible to achieve a system that generalizes well to all kinds of terrain purely by manual parameter tuning, as the search space is prohibitively large.

We seek to learn the optimal trade-offs for foothold selection, using footholds that are demonstrated by an expert. To this end, we formally define the foothold selection problem, describe the process of collecting expert demonstrations, and then introduce an algorithm that learns how to select good footholds.

5.1. Problem Formulation

The state of the robot is defined as the 3D position of all feet, 3D body position and orientation, and the knowledge of which leg is the swing leg. We define the set of n footholds reachable from this state by $\mathcal{F} = \{f_1 \dots f_n\}$. Each of these footholds f_i is described by a feature vector $\mathbf{x}_i \in \mathbb{R}^d$. We define this feature vector $\mathbf{x}_i$ as being composed of two groups: terrain features, which encode information about the terrain at the foothold, and pose features that quantify all other relevant information such as progress towards the goal, stability margins, and collision margins.

We define a reward function R over footholds as a weighted linear combination of the features:

$$R(f_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector. This reward function can then be used to rank all of the footholds in $\mathcal{F}$, and the one with the highest reward selected as the target foothold.[2]. The task of learning the trade-offs for foothold selection has thus been reduced to learning the weight vector $\mathbf{w}$.

5.2. Collecting Expert Demonstrations

An expert is shown logs of runs executed by the robot on different kinds of terrain. They inspect each run for sub-optimal foot placement choices made by the robot, and in each of those situations, labels the correct foothold f_c that they think is the best greedy choice from among the entire set of reachable footholds[3] $\mathcal{F}$. Figure 7(a) shows a screenshot of our teaching interface, where the red ball depicts the sub-optimal foot placement choice made by the robot, and the white ball below shows the optimal foot placement chosen by the expert.

5.3. Learning from Expert Demonstrations

When the expert labels the foothold f_c from the set $\mathcal{F}$ as the optimal foothold, they are implicitly providing information that the reward for the chosen foothold f_c is better than the reward for all others:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_c > \mathbf{w}^{\mathrm{T}}\mathbf{x}_i \quad \forall i \in \mathcal{F};\ i \neq c. \tag{2}$$

This corresponds to the widely studied rank learning or preference learning problem in the machine learning literature (Cohen et al. 1999). This problem can be converted into a binary classification problem (Joachims 2002). Rearranging Equation (2), we obtain

$$\mathbf{w}^{\mathrm{T}}(\mathbf{x}_c - \mathbf{x}_i) > 0 \quad \forall i \in \mathcal{F};\ i \neq c. \tag{3}$$

The optimum value of $\mathbf{w}$ could be found by defining a suitable loss function that penalizes values of $\mathbf{w}^{\mathrm{T}}(\mathbf{x}_c - \mathbf{x}_i)$ that are lower than zero. However, it is observed that such a loss function would be identical to that of a linear binary classifier, in which the inputs are the pairwise difference feature vectors $(\mathbf{x}_c - \mathbf{x}_i)$, all mapped to the target class $+1$. Hence, the weights $\mathbf{w}$ can be obtained by running a linear classifier[4] such as Support Vector Machines (SVM) or logistic regression (LR) on the pairwise difference feature vectors $(\mathbf{x}_c - \mathbf{x}_i)$. The resulting weight vector $\mathbf{w}$ is used to evaluate the reward R in Equation (1) on all candidate footholds $f_i \in \mathcal{F}$. The ranking imposed upon $\mathcal{F}$ by the reward R will satisfy all of the training data in Equation (2), if and only if the expert's ranking function is truly a linear function of the features. We show in the following section that this is usually not the case, especially with conventional feature representations of footholds.

5.4. Terrain Templates

Some of the terrain features previously used for the foothold selection problem are slope, curvature, and standard deviation of the terrain on various spatial scales (Kolter et al.

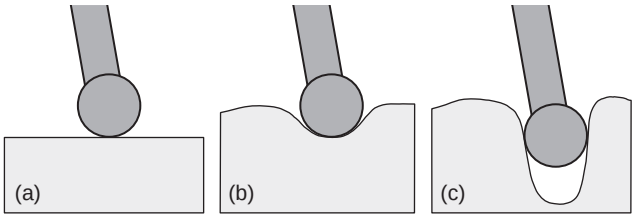

Fig. 6. Three footholds with increasing values of curvature: (a) flat terrain; (b) terrain with mild curvature, preferable to flat terrain since it prevents slipping; (c) terrain with high curvature, which can cause the foot to get stuck, thus is less preferable than (a) or (b).

2007, 2008). However, since the rewards are defined to be linear in the features, such a feature set can only represent more complex decisions by careful engineering of non-linear features. For example, Figure 6 shows three footholds with increasing levels of curvature. The foothold illustrated in Figure 6(b) is preferable to flat terrain (Figure 6(a)), because it prevents foot slippage. The foothold illustrated in Figure 6(c), however, is not, because the foot could get stuck in it. Such a ranking is hard to achieve with a reward function that is a linear function of curvature.

Although this specific example could possibly be handled by hand-crafting additional features, one can imagine that many more such situations exist which cannot be foreseen. The problem thus requires a much more general solution.[5]

We propose the concept of a *terrain template*, which is a discretized height map of the terrain in a small area around the foothold. One could imagine an approach in which the robot maintains a library of such templates, each associated with a reward value. Subsequently, during execution, each candidate foothold can be assigned the reward from the closest matching template in the library (using an arbitrary similarity measure). In the context of Figure 6, the robot would store a template for each of the three footholds shown, along with an appropriate reward for each. Assigning the highest reward to the template for Figure 6(b) would allow the robot to select that foothold over the others.

Manual creation of such a library of templates would be too time consuming, and it would be nearly impossible to attain good generalization performance, since the rewards for each template would have to be tuned carefully. Hence, we propose an algorithm that uses expert-demonstrated footholds to simultaneously learn a small set of templates, and a foothold ranking function that uses these templates. We first describe our methodology for template extraction from expert demonstrations, followed by the template learning algorithm.

5.5. *Template Extraction*

From each foothold demonstration $\langle \mathcal{F}, f_c \rangle$ made by the expert, we extract a set of templates on multiple scales. Figure 7(b) shows the three scales of templates that were

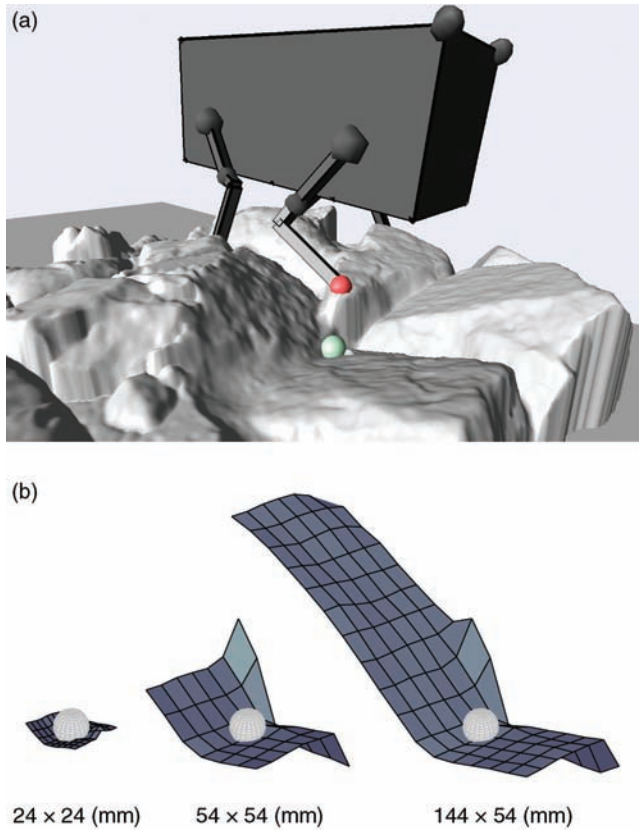

Fig. 7. (a) Teaching interface used to demonstrate footholds. Red ball at the foot indicates the chosen (dangerous) foothold, light green ball below indicates the demonstrated optimal (safe) foothold. (b) Terrain templates in multiple scales extracted from the demonstrated foothold. The white spheres indicate the position of the foot on each template.

extracted from the demonstrated foothold (light green ball in Figure 7(a)). The scales that we use are dictated by the geometry of our quadruped robot, and are designed to independently capture different properties of the terrain that make up its reward function. The smallest scale (24 mm $\times$ 24 mm) is of the order of the size of the foot, and encodes information about surface friction properties and micro-features that define the slip characteristics of the foothold. We intend to capture information about clearance from obstacles and drop-offs using the medium scale (54 mm $\times$ 54 mm). Finally, the large scale (144 mm $\times$ 54 mm) accounts for potential shin and knee collisions. The representation of different properties of the terrain using multiple scales at different resolutions prevents the problem of over-fitting that can occur if only the largest scale is used at high resolution.

Templates on all scales are also extracted from all the other reachable footholds in the set $\mathcal{F}$, for every expert demonstration. This creates a large library of templates representing different kinds of terrain. This comprises the input to the following template learning algorithm, which selects a small subset of these templates and learns a foothold ranking function using them.

5.6. Template Learning

Each template in the library contributes a feature to the feature vector $\mathbf{x}_i$ of a foothold f_i. The feature value represents the similarity between the template and the candidate foothold. We choose a radial basis function similarity measure: a negative squared exponential kernel centered at the template, and evaluated at the candidate foothold. The feature value is given by

$$x = \exp\left(-h \sum_{i=1}^{n} (t_i - c_i)^2 \right), \quad (4)$$

where x is the value of the feature, h is the bandwidth parameter of the kernel,[6] n is the number of terrain points in the discretized height map of the template, t_i is the ith height value of the template, and c_i is the ith height value of the candidate foothold.

This feature set forms the input to the algorithm described in Section 5.3, that learns a ranking function for footholds from expert demonstrations. This is done by converting the expert ranking data into a problem of linear binary classification, and using the resulting weights as a ranking function. However, since we have a potentially huge template library, selecting a small subset of these templates for use in the final ranking function is important for two reasons. First, the use of the entire template set would allow the learning algorithm to trivially overfit all of the training data, resulting in poor generalization. Secondly, the similarity measure for templates needs to be evaluated in real-time while the robot is walking, and these computations can be prohibitively expensive for a large template library.

We combine the steps of template subset selection and weight learning by using a linear classifier that promotes sparsity in feature space. We run an efficient implementation of l_1-regularized logistic regression (LR) (Koh et al. 2007) on the pairwise difference feature vector data $(\mathbf{x}_c - \mathbf{x}_i)$; l_1- regularized LR is a linear classifier that can be defined as an optimization problem to minimize the following cost function:

$$J = \sum_{i=1}^{m} -\log\left(\frac{1}{1 + \exp(-\mathbf{w}^{\mathrm{T}} \mathbf{x}_i y_i)} \right) + \lambda \|\mathbf{w}\|_1, \quad (5)$$

where m is the number of training examples, $\mathbf{x}_i \in \mathbb{R}^d$ is the ith input vector, $y_i \in \{-1, 1\}$ is the label of the ith input vector, $\mathbf{w} \in \mathbb{R}^d$ is the weight vector, and λ is the regularization parameter. The predicted label y_{test} for an input $\mathbf{x}_{\text{test}}$ is obtained using

$$y_{\text{test}} = \text{sgn}(\mathbf{w}^{\mathrm{T}} \mathbf{x}_{\text{test}}). \quad (6)$$

The second term in Equation (5) is a regularization term that penalizes the l_1 norm of the weight vector. The use of an l_1 regularization term has been known to produce sparse solutions, i.e. a weight vector $\mathbf{w}$ which contains very few non-zero values (Ng 2004). It has been shown that l_1-regularized LR can outperform l_2-regularized LR, especially in cases where the number of features is much larger than the number of data points (Ng 2004). In our case, each expert demonstration produces a large set of training examples for the classifier. However, data from footholds that are very close to each other is likely to be very similar, so the true number of unique training examples that convey novel information is usually much lower.

The regularization parameter λ can be viewed as a control for the desired amount of sparsity: higher values of λ typically produce sparser weight vectors. We fix the value of λ by searching through a range of values and picking the one that minimizes the cross-validation training error.

The sparse weight vector $\mathbf{w}$ that we obtain from the l_1-regularized LR allows us to discard all of the templates from the library which have a zero weight. The reward for each foothold can now be evaluated by calculating the feature values for templates that remain in the library using Equation (4), and evaluating the reward function in Equation (1) using the learnt weight vector $\mathbf{w}$.

5.7. Results

In order to evaluate the performance of our template learning approach, we collected around 100 expert foothold demonstrations across different kinds of terrain of varying difficulty levels. Three different foothold reward functions were learnt using the following feature sets: (a) multi-scale terrain features, (b) multi-scale terrain templates and (c) terrain templates + terrain features. Twenty six 26 pose features were also used along with each of the above feature sets. The pose features included measures of progress towards the goal, stability margins, reachability of the foothold, differences in leg heights, and knee and body clearance from the terrain.

The terrain features we used (on each scale) were slope and curvature along the x- and y-axes,[7] and standard deviation of the terrain. This resulted in a total of nine features being used per length scale. The same three length scales were used for both feature computation and template extraction, and are depicted in Figure 7(b). The features and templates are mirrored in the y direction for the left and right legs, but independent functions were learnt for front and hind legs as their properties tend to be quite different.

A foothold ranking function was learnt for each of the three feature sets mentioned above, using the algorithms described in Sections 5 and 5.4. The l_1 regularization parameter λ was set individually in each case by choosing the value that minimized cross-validation error on the training data. Table 1 shows the number of templates selected on each scale, with and without the use of terrain features. Interestingly, almost no templates were needed on the smallest scale when features were used, however, most of the templates on the medium and large scales were still required. This suggests that the expert's ranking function contained a significant amount of non-linearities in the

Table 2. Success rates and average slip at footholds using different feature sets. Experimental details can be found in Section 5.7.

Method	Success rate (out of 21 runs)	Average slip on successful runs (mm)
Features only	47.6%	24.8 ± 5.9
Templates only	76.2%	20.2 ± 5.1
Templates + features	100.0%	17.3 ± 3.3
Baseline (on flat terrain)	100.0%	13.4 ± 0.4

larger scales that terrain features were unable to capture. At the smallest scale, the information contained by the templates seems to be identical to that of the features computed at this scale.

We measured locomotion performance using two metrics: success rate and average slip experienced by the robot at each foothold. A run was deemed a success if the robot crossed the terrain and reached the goal without falling over. Slip was measured as the distance between the position of a foot at touchdown and the position of the same foot before it swings again, i.e. one full walking cycle later. We averaged the slip only over successful runs, since including the slip experienced on runs that failed would render the statistic meaningless. Table 2 shows the results that were obtained by performing 21 runs using the three different ranking functions learnt from the three feature sets. These runs were performed on a test terrain module which the robot was not trained on. The use of templates is seen to improve performance over that of terrain features, in terms of both success rate and slip. It is also observed that combining templates and features results in the most robust performance and lowest average slip. The baseline slip performance of the robot walking on flat terrain is also shown, as a lower bound to the achievable amount of slip on our experimental setup.[8]

The results achieved using a combination of templates and features suggest that achieving broad generalizations using heuristic features, while simultaneously representing exceptions to those rules by using templates results in a high-performance system. We stress the fact that since the templates are learnt automatically, strong regularization during the learning process is a key component in achieving generalization and preventing overfitting to the training data.

5.8. Discussion

Our technique for learning good foot placement is inherently a *supervised* learning approach, wherein an expert demonstrates good footholds and the algorithm learns how to choose footholds with similar characteristics. This is in contrast to *reinforcement* learning (RL) approaches, where the robot would learn to select good footholds based on its own past successes and failures. The RL technique can possibly result in performance superior to that achievable by learning from expert demonstrations, since the expert is not necessarily always optimal in their choices. Indeed, we have experimented with RL approaches in the early stages of the project. As promising as they might be, they require a very large number of trials over the terrain in order to learn a reasonable foothold selection function. This tends to be quite taxing on the robot hardware. When it comes to very difficult terrain, where good footholds are few and far between, the probability of a successful trial during RL exploration is low, making learning very difficult. Our supervised learning approach learns much faster with the help of direct feedback provided by the expert.

6. ZMP-constrained Body Trajectory Generation

The body trajectory generator uses the next four footholds (as planned by the footstep planner) and creates a body trajectory. The generated trajectory is smooth, stable with respect to the 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 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 (Vukobratovic and Borovac 2004) as our stability criterion. The use of the ZMP criterion is quite popular, especially in the field of biped and humanoid locomotion (Sugihara et al. 2002; Kajita et al. 2003). In most of these works, the desired ZMP trajectory during a leg swing motion is either fixed to be in the center of the supporting foot, designed by a user, or derived from a dynamics model, such as the linear inverted pendulum model (Kajita et al. 2002). This desired ZMP trajectory is then tracked using specially designed controllers. In the case of biped and humanoid robots, the support polygon tends to be quite small in relation to the height of the COG, and there is not much freedom in designing a ZMP trajectory within that small area. In contrast, our LittleDog robot has a relatively low COG in comparison with the support triangle created by three stance legs. In the algorithm that we present below, our intention is to make use of this large area in order to generate smooth and optimal COG motions that are stable according to the ZMP criteria. In contrast to earlier work, we generate both the desired ZMP and COG

Fig. 8. Sequence of snapshots of the quadruped robot LittleDog crossing the test terrain after template learning (left to right, top to bottom).

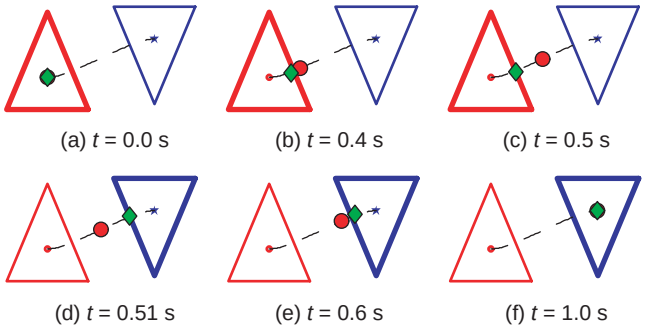

Fig. 9. Snapshots from a Zero-Moment Point (ZMP)-stable body trajectory generated by our optimizer. The two support triangles shown are disjoint because they have been shrunk to accommodate a stability margin. The COG (red circle) moves smoothly between the two disjoint support triangles, while the ZMP (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, which influences the ZMP position according to Equation (14). The discrete jump in the ZMP requires discontinuous accelerations, hence in practice we allow a 50 ms "four-leg support phase" for the ZMP to move smoothly from one triangle to the next.

trajectories in a single optimization problem, as opposed to generating a ZMP trajectory and then attempting to track it.

We represent the trajectory of the COG as a series of quintic spline segments and formulate body trajectory generation as an optimization problem to minimize squared accelerations along the trajectory, subject to ZMP stability constraints. The cost function is quadratic in the spline coefficients, and the constraints are linear. The resulting optimization is a convex 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 during each swing leg movement is split up into n quintic spline segments (we use $n = 3$), each segment being represented by the following:

$$x(t) = at^5 + bt^4 + ct^3 + dt^2 + et + f, \tag{7}$$

where $x(t)$ is the body trajectory along the x-axis as a function of time t, and a–f are the spline coefficients. Equations presented in this section for the x-axis also equally apply to the y-axis, by replacing all occurrences of x with y. In order to obtain an expression for the cost function, Equation (7) is differentiated twice to obtain the expression for the acceleration of the trajectory:

$$\dot{x}(t) = 5at^4 + 4bt^3 + 3ct^2 + 2dt + e, \tag{8}$$

$$\ddot{x}(t) = 20at^3 + 12bt^2 + 6ct + 2d. \tag{9}$$

Squaring and integrating the function for acceleration, we get the cost function for a single spline segment, which can be written in matrix form as:

$$\int_0^T \ddot{x}^2(t)\, dt = \mathbf{q}^T \mathbf{G} \mathbf{q}, \tag{10}$$

where

$$\mathbf{q} = \begin{bmatrix} a & b & c & d \end{bmatrix}^T,$$

$$\mathbf{G} = \begin{bmatrix} \frac{400}{7}T^7 & 40T^6 & \frac{120}{5}T^5 & 10T^4 \\ 40T^6 & \frac{144}{5}T^5 & 18T^4 & 8T^3 \\ \frac{120}{5}T^5 & 18T^4 & 12T^3 & 6T^2 \\ 10T^4 & 8T^3 & 6T^2 & 4T \end{bmatrix},$$

and T is the duration of the spline segment. The cost function is clearly quadratic in the coefficient vector $\mathbf{q}$, and the matrix $\mathbf{G}$ is symmetric and positive definite. The final cost function used in the optimization is the sum of individual cost functions for each spline segment.

Double differentiability of the body trajectory is important when using inverse dynamics control, to guarantee continuous motor torques. Each spline segment on its own, being a fifth-order polynomial, is twice differentiable. In order to guarantee this property in between spline segments, constraints are constructed by equating the position, velocity and acceleration (from Equations (7), (8), and (9)) at the junction of two neighboring spline segments:

$$x_i(T_i) = x_{i+1}(0), \tag{11}$$

$$\dot{x}_i(T_i) = \dot{x}_{i+1}(0), \tag{12}$$

$$\ddot{x}_i(T_i) = \ddot{x}_{i+1}(0), \tag{13}$$

where subscripts refer to the index of the spline segment. These constraints are observed to be 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 (Kajita et al. 2003) as an approximation of the LittleDog robot, which has a heavy body relative to its legs. This model relates the COG and ZMP positions according to

$$x_{\mathrm{ZMP}} = x_m - \frac{z_m \ddot{x}_m}{\ddot{z}_m + g}, \tag{14}$$

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 Equation (14).

At every instance 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 points 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 one side of it. For example, if one of the lines is given by $px + qy + r = 0$, its corresponding constraint is $px + qy + r > 0$ (the sign of the inequality may vary depending on the order of points used to construct the line equation). In order to represent this constraint for all times, we discretize the trajectory at the control interval of LittleDog (10 ms), and express the ZMP constraints at each of these times, by substituting the expressions for $x(t)$ and $y(t)$ from Equation (7). Clearly, these constraints are also linear in the spline coefficients, since p, q, and r are constants, and $x(t)$ and $y(t)$ are linear.

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 (see http://quadprog.sourceforge.net/) to perform the optimization, which is typically completed in 30 ms. Situations may arise where the optimization fails, i.e. there is no feasible trajectory that can satisfy all of 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 Figure 9, 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 (50 ms) 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 preferable for our inverse dynamics controller, as described in Section 4.9.

6.1. Discussion

The ZMP criterion is only applicable in cases where the foot contact points are coplanar and have unidirectional normal forces. In addition, we assume that the friction is high and that the contact points do not slip. Modifications of the ZMP stability criterion to take these factors into account are discussed at length in Stonier and Kim (2006). In our work, we simply ignore these effects, accommodating errors in the ZMP by shrinking support triangles by a stability margin. This parameter is empirically tuned for good performance over irregular terrain.

7. Inverse Dynamics Control

In the following we describe how we use an inverse dynamics control law to achieve active compliance. This element makes the robot more robust towards perturbations and unknown terrain.

7.1. Floating-base Inverse Dynamics Control

While inverse dynamics control for a fixed-base robot, such as an industrial manipulator, is textbook knowledge (Sciavicco and Siciliano 2000), to develop a general inverse dynamics control law for a floating-base system such as a legged robot is ongoing research. Practical application of floating-base inverse dynamics has been hindered by the following issues: (a) dependence on precise dynamics models, amplified by numerical problems that can arise due to matrix inversions, in particular the inversion of the rigid body dynamics (RBD) inertia matrix (Nakanishi et al. 2008), (b) external forces need to be measured, and (c) no general analytically correct framework for floating-base systems for arbitrary constraints from the environment and closed-loop kinematic chains.

The treatment of floating-base systems has a long history. A very straightforward way to remove the complications associated with the floating base is to directly derive the equation of motion in the constraint space (Sciavicco and Siciliano 2000) and reduce the problem to a fixed-base problem. However, this requires us to re-derive a model for each new contact case, which makes it impractical for legged locomotion and other robotic applications such as mobile manipulation. Floating-base descriptions have been

Table 3. Tracking results and results of robustness step on non-perceived obstacles for different controller types. (HG) high gains, (MG) medium (half) gains, (LG) low (1/6) gains. Control laws: (PD) PD position control only, (F) PD + force control, (F/ID) PD + force and inverse dynamics, (ID) PD + inverse dynamics. Bottom row, average foothold tracking. Top panel: simulation results (Pass/Fail). Bottom panel: real robot, number of passed trials out of 10.

(A) simulation 250209

obst. height	HG				MG				LG			
[cm]	PD	F	F/ID	ID	PD	F	F/ID	ID	PD	F	F/ID	ID
1.0	P	P	P	P	P	P	P	P	X	X	P	P
2.0	P	P	P	P	P	P	P	P	X	X	P	P
3.0	P	P	P	P	P	P	P	P	X	X	P	P
4.0	F	F	P	F	F	P	P	P	X	X	P	P
5.0			F	F	F	F	P	F	X	X	P	P
6.0							F	F	X	X	P	F
6.5									X	X	P	F
7.0									X	X	F	F
avg. tracking [cm]	0.12	0.11	0.25	0.21	0.39	0.34	0.29	0.29	fail	fail	0.63	0.39

(B) real world 270209

obst. height	HG				MG				LG			
[cm]	PD	F	F/ID	ID	PD	F	F/ID	ID	PD	F	F/ID	ID
2.0	10	10	10		10	10			X	X		
4.0	0	0	6	7	1	0	9	10	X	X	10	10
6.0			0	0			0	0	X	X	9	9
7.0									X	X	2	2
avg. tracking [cm]	0.92	0.90	0.50	0.51	1.22	1.49	0.85	0.77	fail	fail	2.37	2.69
std	0.47	0.46	0.21	0.20	0.82	0.86	0.41	0.37			1.31	1.40

introduced to model space systems (e.g. satellites) and derive controllers for such systems (Dubowsky et al. 1999). However, these systems usually do not make contact with the inertial frame and are therefore not affected by the problems associated with switching contacts.

In a recent development (Mistry 2009), we demonstrated how we can address these issues. As it turns out, our solution, while still dependent on a model, is less susceptible to modeling errors as it avoids inversion of the RBD inertia matrix (e.g. such as in Sentis (2007))), such that it also alleviates issue (a). External forces need not be measured, resolving issue (b). While our approach is not completely worked out for all contact cases, (c) is partially taken care of by planning constraint satisfying reference trajectories.

Our way of computing inverse dynamics control for floating-base systems can be applied to arbitrary robots with multiple and dynamically changing constraints. Owing to the problems that are caused by working with measured contact forces directly, we have developed an approach that does not require the use of the measured contact forces in the calculation of the inverse dynamic torques. This solution is accomplished by computing the inverse dynamics torques in the reduced-dimensional null-space of the constraints, as realized by an orthogonal decomposition of the constraint Jacobian. A similar approach is shown in Aghili (2005). In our contribution we focus on the application of the projection-based inverse dynamics to improve the performance of walking systems in rough terrain.

In the following we shortly review the central results of our floating-base inverse dynamics framework. For further details please refer to Mistry (2009).

7.2. Floating-base Dynamics

The configuration of the full floating-base system is described as

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_r^{\mathrm{T}} & \mathbf{x}_b^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{15}$$

where $\mathbf{q}_r \in \mathbb{R}^n$ is the joint configuration of the rigid body robot with n joints and $\mathbf{x}_b \in \mathbb{R}^6$ is the position and orientation of the coordinate system attached to the robot base, and measured with respect to an inertial frame (cf. Mistry (2009) for further details). When the robot is in contact with the environment, the equations of motion with respect to an inertial frame are given by

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^{\mathrm{T}}\tau + \mathbf{J}_C^{\mathrm{T}}(\mathbf{q})\,\lambda \tag{16}$$

with variables defined as follows:

- $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n+6\times n+6}$: the floating-base inertia matrix;
- $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n+6}$: the floating-base centripetal, Coriolis, and gravity forces;
- $\mathbf{S} = \begin{bmatrix} \mathbf{I}_{n\times n} & \mathbf{0}_{n\times 6} \end{bmatrix}$: the actuated joint selection matrix;
- $\tau \in \mathbb{R}^n$: the vector of actuated joint torques;
- $\mathbf{J}_C \in \mathbb{R}^{k\times n+6}$: the Jacobian of k constraints;
- $\lambda \in \mathbb{R}^k$: the vector of contact forces.

In order to be able to derive an inverse dynamics control law that is not dependent on measured forces we make the following assumptions:

1. $\dot{\mathbf{x}}_C = \ddot{\mathbf{x}}_C = \mathbf{0}$ is maintained, i.e. there is sufficient friction and stiffness at the constraint locations to prevent motion.

2. The system is not over-constrained: The matrix $\mathbf{J}_C$ should remain full row rank, i.e. every constraint can be satisfied simultaneously.
3. The system is sufficiently constrained to eliminate under-actuation: if we divide the constraint Jacobian into its parts ($\mathbf{J}_C = \begin{bmatrix} \partial\mathbf{x}_C/\partial\mathbf{q}_r & \partial\mathbf{x}_C/\partial\mathbf{x}_b \end{bmatrix}$), the constraint Jacobian related to base motion ($\partial\mathbf{x}_C/\partial\mathbf{x}_b$), must have a rank equal to six.

It is important to note that while these assumptions are needed to ensure that the calculated torques are *analytically* correct they are not numerically critical. This means even when some of the assumptions are violated (which is likely to happen on a real system), the algorithms are still numerically stable. This means while the torques will not be 100% accurate they cannot grow to infinity. They remain meaningful and useful for practical purposes. Refer to Mistry et al. (2008) for further discussion of these assumptions. Such graceful degradation with respect to the violation of the assumptions together with avoiding the inversion of inertia matrices makes our method applicable to *real* systems; this has been shown in recent contributions (Buchli et al. 2009) and will be illustrated in the results section.

7.3. QR Decomposition of the Constraint Jacobian

The underlying idea of computing our inverse dynamics control law is to separate constrained from unconstrained dynamics. To this end the QR decomposition of the contact Jacobian $\mathbf{J}_C$ is computed

$$\mathbf{J}_C^{\mathrm{T}} = \mathbf{Q}\begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}, \tag{17}$$

where $\mathbf{Q}$ is orthogonal ($\mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$), and $\mathbf{R}$ is an upper triangle matrix of rank k.

7.4. Inverse Dynamics Law as Projection of the Floating-base Dynamics

By providing $\mathbf{Q}$ the $\mathbf{QR}$ decomposition yields a coordinate transform into a coordinate system in which the constrained and unconstrained dynamics naturally separate along axes of the coordinate system, as can be seen by multiplying (16) by $\mathbf{Q}^{\mathrm{T}}$:

$$\mathbf{Q}^{\mathrm{T}}\left[\mathbf{M}\ddot{\mathbf{q}} + \mathbf{h}\right] = \mathbf{Q}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}\tau + \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}\lambda. \tag{18}$$

Therefore we can select the unconstrained dynamics with a simple selection matrix $\mathbf{S}_u = \begin{bmatrix} \mathbf{0}_{(n+6-k)\times k} & \mathbf{I}_{(n+6-k)\times(n+6-k)} \end{bmatrix}$. We are therefore left with the unconstrained dynamics which is not dependent on the contact forces

$$\mathbf{S}_u\mathbf{Q}^{\mathrm{T}}\left[\mathbf{M}\ddot{\mathbf{q}} + \mathbf{h}\right] = \mathbf{S}_u\mathbf{Q}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}\tau \tag{19}$$

Solving for τ leads to the inverse dynamics

$$\tau = \left(\mathbf{S}_u\mathbf{Q}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}\right)^{+}\mathbf{S}_u\mathbf{Q}^{\mathrm{T}}\left[\mathbf{M}\ddot{\mathbf{q}}_d + \mathbf{h}\right] \tag{20}$$

where $(\cdot)^+$ is the right pseudo-inverse ($A^+ = A^{\mathrm{T}}\left(AA^{\mathrm{T}}\right)^{-1}$).

Equation (20) does not depend on the contact forces and produces the inverse dynamics torques that will realize the desired joint accelerations $\ddot{\mathbf{q}}_d$. This control law is just a projection of the floating-base inverse dynamics (the term in square brackets), which can be computed with efficient algorithms that scale linearly with the number of the DOFs of the robot (Featherstone 2007).

Please refer to Mistry (2009) for a more extensive development and discussion of the presented floating-base inverse dynamics law.

7.5. Control Law and Dynamics Model

To realize the inverse dynamics controller we use a standard control law that combines the feed-forward torque computed based on the desired acceleration $\ddot{\mathbf{q}}_d$ and the current state of the robot $\mathbf{q}, \dot{\mathbf{q}}$ with a negative feedback correction computed via a standard PD controller:

$$\tau = \tau_{\mathrm{ID}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_d) - \mathbf{K}_P\left(\mathbf{q} - \mathbf{q}_d\right) - \mathbf{K}_D\left(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d\right), \tag{21}$$

where τ_{ID} is the inverse dynamics feed-forward term computed using Equation (20), and $\mathbf{K}_P > 0$ and $\mathbf{K}_D > 0$ are the proportional and differential gain, respectively. The dynamics model uses a mix of CAD-based data and estimated parameters.

7.5.1. Force Control We can use the inverse dynamics to predict the contact forces at the stance feet. The calculation of the forces are a direct consequence of our general inverse dynamics formulation, and therefore works for all contact cases and takes dynamics into account. It therefore is more general than earlier methods to address force control for legged robots (e.g. Dubowsky et al. 1999). The contact forces can be calculated as (Mistry 2009):

$$\lambda = \mathbf{R}^{-1}\mathbf{S}_c\mathbf{Q}^{\mathrm{T}}\left[\mathbf{M}\ddot{\mathbf{q}}_d + \mathbf{h} - \mathbf{S}^{\mathrm{T}}\tau\right], \tag{22}$$

where

$$\mathbf{S}_c = \begin{bmatrix} \mathbf{I}_{k\times k} & \mathbf{0}_{k\times(n+6-k)} \end{bmatrix}. \tag{23}$$

Since we do not rely on measured contact forces to compute the inverse dynamics, we can instead use the predicted contact forces (Equation (22)) to add a force control term to our controller:

$$\tau_F = \mathbf{J}_F^{\mathrm{T}}\left(\mathbf{F}_m - \lambda\right), \tag{24}$$

where $\mathbf{F}_m$ are the measured foot forces and $\mathbf{J}_F^{\mathrm{T}}$ is the Jacobian transpose of the foot forward kinematics. This torque is simply added to the one calculated via the ID/PD control law (Equation (21)).

As a result, in the case of unexpected terrain contact, this term will counteract the PD controller, thus the robot will

not continue to push into the terrain with all of the force. This active compliance makes the behavior more robust. The effects of the force controller on the robustness of the system in the face of uncertain terrain is discussed in Buchli et al. (2009).

7.6. Experimental Results

In the following we compare the performance of different versions of the presented controller with varying gain settings on different setups. We present results both from simulation as well as from the real robot.

7.6.1. Joint Space Tracking Results In order to show that the inverse dynamics torque is the most important contribution to the control torque in nominal behavior, in Figure 10, we show a representative example of the tracking of the knee joint along with the total motor command. Typically, it can be seen that the total motor command is mostly accounted for by the feed-forward contribution. However, while in theory the feed-back command should be almost negligible, in the real robot we have a continued clear contribution of the feedback controller. This is due to the imperfect dynamics model of the robot, and, therefore, the feed-forward command is not able to track the desired trajectory perfectly. It is worth noting that the floating-base inverse dynamics controller has a graceful degradation despite a rather imprecise model of the inverse dynamics parameters. Other approaches, which require inversion of the RBD inertia matrix can create significant problems in face of modeling errors (Nakanishi et al. 2008).

7.6.2. Improvement of Robustness Over Non-perceived Obstacles A total of 12 different controllers were tested in a physics based simulator (Schaal 2009). Three different gain settings (HG) high gains, (MG) medium gains (half of the HG settings) and (LG) low gains at 1/6 of the HG settings. All of these three gain settings were run with four different control laws: (PD) PD position control only, (F) PD + force control, (F/ID) PD + force and inverse dynamics, and (ID) PD + inverse dynamics.

All controllers were subjected to an unperceived obstacle of variable height (cf. Figure 11) from 1 to 7 cm (corresponding to up ∼50% of the leg length). Table 3 lists the results. A P denotes passing the obstacles without falling and F denotes falling over and therefore failure. The experiments were repeated 10 times and despite the simulator having stochastic elements due to the "penalty method" employed to model ground contacts, the results were consistently either pass or fail for a given setup and obstacle height.

While lowering the gains (MG) already gives higher robustness, it is not strictly mandatory to use inverse dynamics with these gain settings. The robot fulfills the task pretty well with PD control only. However, in the case of LG, the PD only (and PD/F) control is not able to track well enough that the robot is even able to take a single step (cf. tracking results). To lower the gains that much inverse dynamics is the crucial element to achieve even basic task fulfillment. From the last two columns in the table we see that the lower gains achieve the highest tolerance for non-perceived obstacles, with the combination of ID/F control giving the best performance.

As outlined above in our case reaching the footholds is of utmost importance. Hence, the tracking quality was assessed by evaluating the average distance of the swing foot from the desired foothold at touch-down and the results are listed in the last row in Table 3.

Looking at the tracking results it is clear that with reducing the gains, as expected, we reduce the accuracy but at the same time gain a large amount of robustness in the presence of not or wrongly perceived obstacles. The reduction of foothold tracking is still acceptable. The benefits in robustness thus justify the use of such a low-gain inverse dynamics and force controller.

7.6.3. Real-world Tests The same tests were repeated on the real robot. Table 3(b) shows the results. The results match the predictions of the simulation very well. Again, the LG-F/ID controller is the most robust. However, in contrast to the simulation the LG-ID controller seems to perform equally well as the combined LG-F/ID controller. The tracking performance degrades more significantly in the low-gain condition, although increased robustness is still observed in the same way as in the simulation tests. It should be noted that our inverse dynamics model was not of high quality: it was partially derived from CAD data, and partially from parameter estimation techniques, and we noticed various deficiencies in the quality of the model. Future work will address how to improve this issue, and tracking performance in low-gain control will be improved.

In order to demonstrate the usefulness of the LG-F/ID controller in more realistic scenarios we ran the real robot over non-perceived scattered obstacles of 2–4 cm height and a non-perceived rock board (Figure 11). In such terrain the HG-PD controller fails while the LG-F/ID controller create enough robustness for successful completion of the runs (see Extension 1). These results show that the efficiency of ID/LG also holds on the real robot despite an imperfect model.

7.7. Robust Walking through Ill-perceived Rough Terrain

Now we present the performance of four different controllers for ZMP walking over difficult and partially ill-perceived terrain. The four different controllers that have been used are: (1. PD HG) Full gain PD control, no inverse dynamics, no force control; (2. ID HG) Inverse dynamics and force control, PD control at full gain; (3. ID MG) inverse dynamics and force control, PD control at half gain; (4. ID LG) inverse dynamics and force control, PD control at quarter gains.

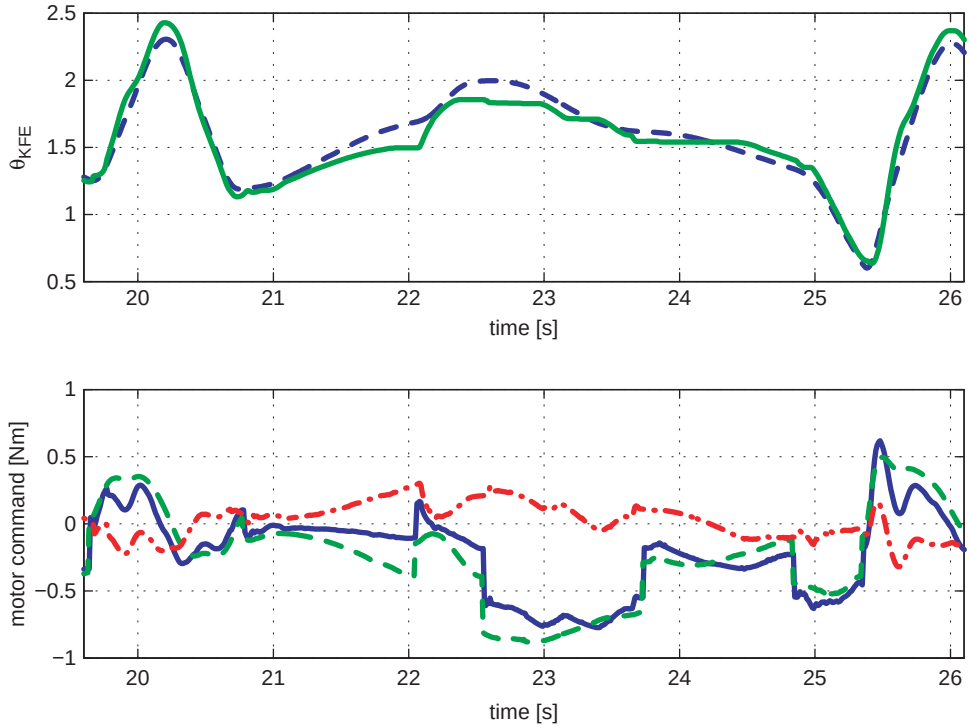

Fig. 10. Top: Tracking of the right hind knee joint in a steady walk. Bottom: The total motor torques τ (blue line) comprising the feed-forward τ_{ID} (green dashed) and the feed-back command τ_{FB} (red dash-dotted). The feed-forward is the major contributor to the motor command most of the time and the feedback torque is comparatively very low. The effects of the constraint switches are clearly visible as they create discontinuities in τ_{ID}.

7.7.1. COG/ZMP Tracking In Table 4, we list the tracking results for the different controllers on flat ground. Flat ground allows for proper comparison between the results and to investigate on nominal performance (i.e. not investigating disturbance rejection of the controller). As can be seen, the addition of the ID component and keeping the high gains improves the tracking. However, this does not yet lead to the desirable compliance. We therefore investigate two other gain settings, at half and at a quarter of the original gains. As can be seen we can lower the gains to a quarter without getting inferior tracking compared with the high-gain PD only case.

7.7.2. ZMP Walking Over Difficult Terrain Finally, we illustrate the robustness and performance of the controller by testing it on rough terrain. To simulate imprecisely perceived terrain, the height map of the terrain which is akin to a 60 cm $\times$ 60 cm field of round rocks of about 10 cm in diameter has been degraded by adding 1,000 randomly chosen Gaussians with width between 10 and 20 mm and an amplitude between -30 and 30 mm (cf. Figure 12).

We have tested three different controller settings and the results are reported in the last line in Table 4. As can be seen the results are best for the gain set to half of the "normal" position control setting. In the case of quarter gains it can be observed that while the general behavior is very good and especially the impact of slipping and unplanned terrain contacts is greatly reduced, there are cases in which the disturbance rejection, due to the low-gain settings, is too much reduced to keep sufficiently accurate tracking causing the robot to lose stability. These are usually cases where the margins are very small due to climbing up, narrow leg positions, or reaching out which in our experience tend also to be critical for the position control setting.

It should also be noted that often the low-gain settings achieve a higher overall average velocity since the robot spends less time recovering from the effects of terrain impact. However, the systematic study of this effect remains future work. For a visual impression and overview of the results achieved with the locomotion controller, please see Extension 1 (also available at http://www.youtube.com/watch?v=cYo9Whssla8).

8. 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 terrain that we have never seen.

In our lab, we tested our controller on rocky terrain of various difficulty levels, round rocks/pebbles (not moveable), and wavy dunes. The most difficult rocky terrain contains obstacles up to 10 cm (76% of the usable leg length) in

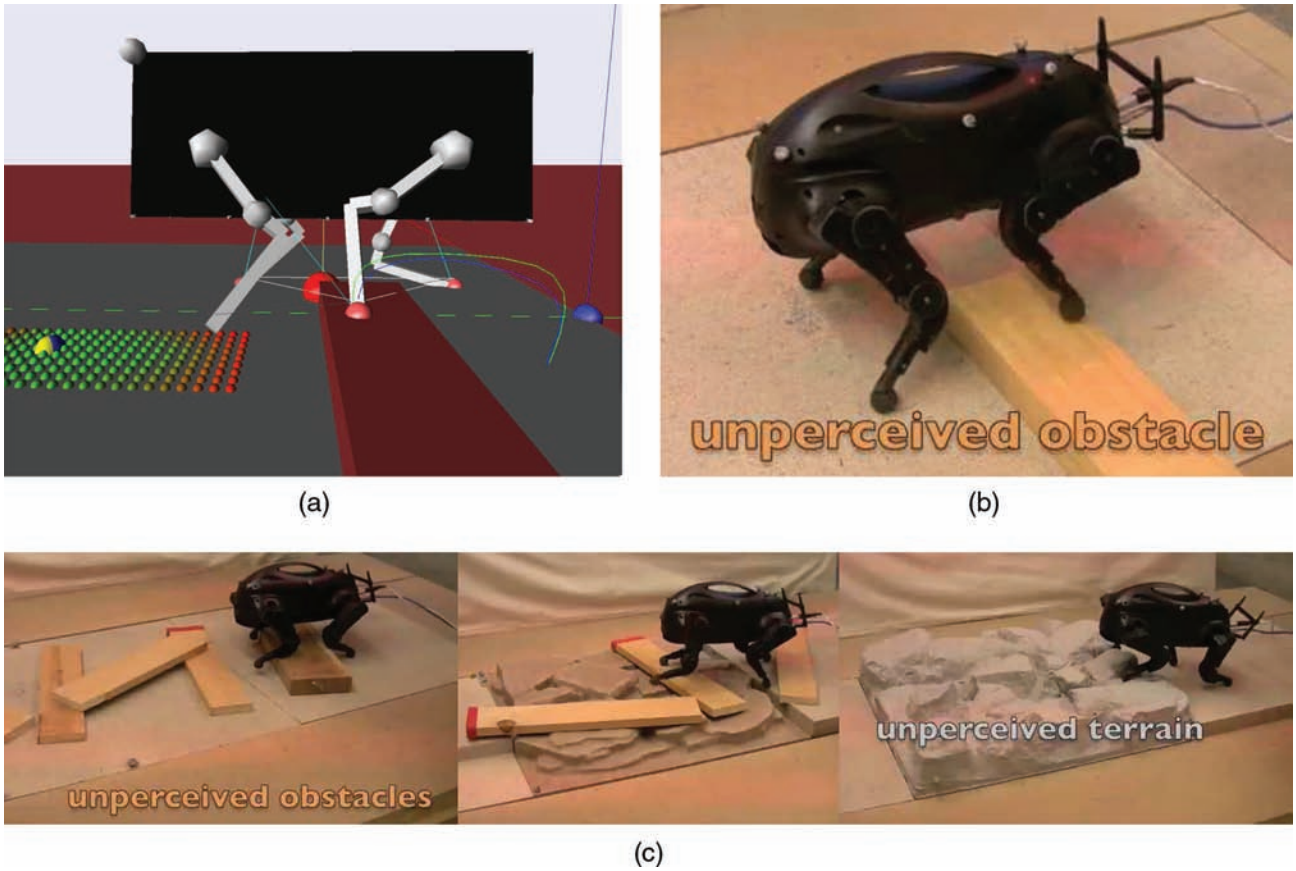

Fig. 11. Setups to test the robustness of the controllers towards non-perceived obstacles: (a) simulation; (b) real world. The brown wooden obstacles are not perceived by the robot. (c) Terrains to assess the performance of the controller on more realistic setups, see Extension 1.

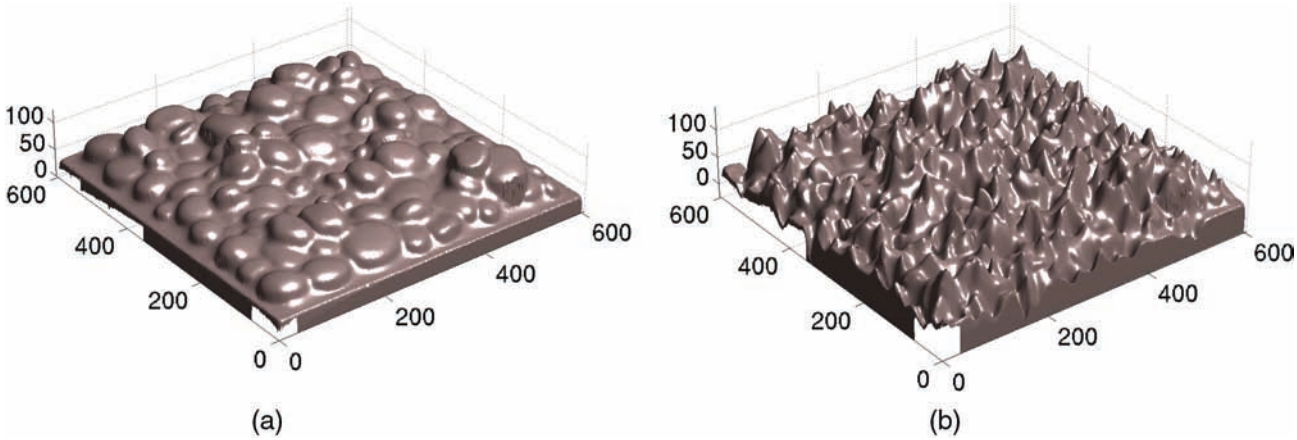

Fig. 12. (a) "Round rocks" test terrain, height map corresponding to the real terrain versus (b) degraded round rocks terrain height map as perceived by the robot.

height. We also tested our software on more structured test scenarios such as 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^{-1} to 13 cm s^{-1} (0.24 to 0.43 body lengths per second). The advantage of our ability to replan quickly in the 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 a footstep plan using the ARA*

Table 4. Tracking performance for the different controllers, the fractions denote the PD gains as a fraction of the PD-only gains: we obtain comparable performance with the PD gains at a quarter of the value used for PD-only control. See the text for further discussion. The last line shows performance on rough terrain walking with impaired perception of the terrain.

	PD HG	ID HG	ID MG	ID LG
Velocity (m s^{-1})	0.097	0.098	0.098	0.095
ZMP tracking, RMSE (m)	0.0176	0.0168	0.0148	0 0174
COG tracking, RMSE (m)	0.0070	0.0045	0.0054	0.0072
Round rocks success rate	3/10	4/10	9/10	3/10

Table 5. Results from final tests of Phase 3 of the Learning Locomotion program. Average speeds are from the best two trials (out of three): individual run speeds were not provided to us. Testing on the first seven terrains was mandatory; these terrains were not shown to us before testing. The last three terrains were optional tests that we chose to demonstrate our performance on.

Terrain	Average speed
Gap	10.7 cm s^{-1}
Barrier	12.1 cm s^{-1}
Rocks	5.2 cm s^{-1}
Sloped Rocks	7.9 cm s^{-1}
Round Rocks	10.0 cm s^{-1}
Logs	7.8 cm s^{-1}
Large Steps	7.2 cm s^{-1}
Wavy Dunes	10.2 cm s^{-1}
Small Steps	10.9 cm s^{-1}
Mixed Steps	7.9 cm s^{-1}

planner, and lunging can again be employed to cross gaps up to 17 cm wide.

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. In addition, a moveable see-saw was set up on the terrain, 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.

Figure 8 shows the robot traversing one of the test terrain modules. Extension 1 (also available at http://www.youtube.com/watch?v=cYo9Whssla8) highlights the key features of our controller, and contains a compilation of video from test runs of the robot on different kinds of terrain.

8.1. Learning Locomotion Test Results

As mentioned in Section 3.3, our control software was subject to additional testing by an external test team, on seven different classes of terrains that have never been shown to us. Table 5 shows the average speeds obtained from the final tests on these seven classes of terrains, as well as three optional terrains. Our controller managed to successfully traverse all classes of terrains in at least two out of three trials, with only one terrain class below the metric speed of 7.2 cm s^{-1}. Although we cannot disclose the performance of the other teams on the program, our team was the only one that crossed at least six out of seven terrains at or above the metric speed. These results serve as a testament to the true generalization ability of our controller.

9. Discussion

Our system is made up of a number of components, as discussed throughout the paper, and depicted in Figure 3. Clearly there are many complex and subtle interactions between these components. In general, when studying the approaches of the other groups involved in the Learning Locomotion project, we see a balance being achieved between planning and control. If planning is performed accurately and at great depth, one can get away with simple high-gain position control to execute this perfect plan. At the other end of the spectrum, one can get away with more approximate planning if the execution of this plan is robust to failure. We believe our approach tends more towards the latter. We use a fast but greedy foot-step planner that is myopic in nature, but its deficiencies are compensated for by using a good swing leg trajectory planner, and executing these plans using our compliant control module.

In a large software and control system like the one we have presented, it is inevitable that there are quite a few parameters that need tuning. In our experience, a large number of the parameters have been of the "set-and-forget" variety, i.e. we tuned them once during the initial development of each module and never needed to revisit them later. The one parameter that does need tuning in our system is the heuristic that defines the duration of a swing leg motion. The locomotion speed of the robot depends primarily on this one parameter, which requires tuning per terrain class for optimal performance. If suboptimal speed were acceptable, as in non-competitive daily life scenarios, no parameter tuning would be required.

Finally, the biggest lesson that we have learnt in the development of this project is that there is no single component or algorithm in our controller that is primarily responsible for achieving good locomotion performance. Each

component plays an equally important role, from selecting the right footholds that do not slip, to compliant model-based control of the robot for increased robustness. It is the combination of all of these techniques that has rendered our approach successful and competitive in the Learning Locomotion project.

10. Conclusion and Future Work

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 achieves high performance 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 competitive, achieving the maximum number of terrains crossed successfully above the metric speed set by the program.

Most of the techniques developed in this work are generally applicable to similar planning and control problems on other locomotion systems. The template-based foothold learning algorithm is a data-driven technique, and does not make assumptions about the number of legs or shape of the foot. The ZMP trajectory optimizer can be applied to any robot which can be approximated by the cart-table model, which has been used before for biped robots (Kajita et al. 2003). Similarly, our floating-base inverse dynamics controller extends to other articulated robots, and has been applied to a simulated biped in other work (Mistry et al. 2010). 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. We also acknowledge the extensive feedback provided by the anonymous reviewers.

This research was supported in part by the DARPA program on Learning Locomotion, National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, IIS-0535282, IIS-1017134, CNS-0619937, IIS-0917318, CBET-0922784, EECS-0926052, NASA grant AC98-516, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, the DARPA program on Advanced Robotic Manipulation, the Army Research Office, the Okawa Foundation, and the ATR Computational Neuroscience Laboratories. JB was supported by a prospective researcher fellowship from the Swiss National Science Foundation.

Notes

1. The robustness afforded by continuous footstep replanning is difficult to quantify, but it is immediately clear to the expert observer that generating a complete footstep plan in advance and attempting to track this plan will fail drastically in the case of slippage or other execution errors
2. Maximizing the immediate reward for the current foothold is an inherently greedy approach. At the expense of higher computation time, one can achieve better performance in practice by choosing the foothold that maximizes a multi-step criterion, i.e. the value function, which is the sum of expected future rewards obtained by choosing a foothold, using standard heuristic graph search algorithms such as A*.
3. Foothold reachability is evaluated by running the pose finder described in Section 4.4, on a grid of footholds around the current leg position.
4. Our choice of a specific classifier is motivated in Section 5.4.
5. Non-linear classifiers, such as the use of radial basis function kernels in SVMs (Scholkopf and Smola 2003) can learn non-linear ranking functions (Joachims 2002), but are still limited by the original feature representation. In addition, the decision function cannot be interpreted as easily as in the linear case.
6. Currently, the bandwidth parameter h of the kernel is tuned manually per length scale. We keep the parameter fixed for all experimental results shown in this paper.
7. We split each slope and curvature feature into two distinct features for the positive and negative directions. The x and y-axes are aligned with the orientation of the robot, i.e. in body coordinates, with x pointing forward and y to the left.
8. The amount of slip experienced on flat terrain might seem rather high. This is largely because the spherical feet on the LittleDog robot roll forward as the robot moves forward, resulting in a systematic change in the perceived foot location.

References

Aghili F (2005) A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation. *IEEE Transactions on Robotics* 21: 834–849.

Bertsekas D, Hager W and Mangasarian O (1999) *Nonlinear Programming*. Belmont, MA: Athena Scientific.

Buchli J, Kalakrishnan M, Mistry M, Pastor P and Schaal S (2009) Compliant quadruped locomotion over rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Byl K, Shkolnik A, Prentice S, Roy N and Tedrake R (2009) Reliable dynamic motions for a stiff quadruped. In *Experimental Robotics*, pp. 319–328.

Cham J, Karpick J and Cutkosky M (2004) Stride period adaptation of a biomimetic running hexapod. *The International Journal of Robotics Research* 23: 141.

Cohen WW, Schapire, RE and Singer Y (1999) Learning to order things. *Journal of Artificial Intelligence Research* 10: 243—270.

Dijkstra E (1959) A note on two problems in connexion with graphs. *Numerische mathematik* 1: 269–271.

Dubowsky S, Sunada C and Mavroidis C (1999) Coordinated motion and force control of multi-limbed systems. *Autonomous Robots* 6: 7–20.

Featherstone R (2007) *Rigid Body Dynamics Algorithms*. New York: Springer.

Fukuoka Y, Kimura H and Cohen A (2003) Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The International Journal of Robotics Research* 22: 187.

Hirose S, Masui T, Kikuchi H, Fukuda Y and Umetani Y (1985) Titan III: a quadruped walking vehicle. In *Robotics Research—The Second International Symposium*, 325–331.

Hodgins J and Raibert M (1991) Adjusting step length for rough terrain locomotion. *IEEE Transactions on Robotics and Automation* 7: 289–298.

Ijspeert A (2008) Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks* 21: 642–653.

Joachims T (2002) Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Kajita S, Kanehiro F, Kaneko K, Fujiwara K, Harada K, Yokoi K, et al. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1620–1626.

Kajita et al 2002: A Realtime Pattern Generator for Biped Walking Kajita, S. and Kanehiro, F. and Kaneko, K and Fujiwara, K and Yokoi, K. and Hirukawa, H. IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION 2002, VOL 1, pages 31–37 Publisher: IEEE

Kalakrishnan M, Buchli J, Pastor P, Mistry M and Schaal S (2010) Fast, robust quadruped locomotion over challenging terrain. In *IEEE International Conference on Robotics and Automation*.

Kalakrishnan M, Buchli J, Pastor P and Schaal S (2009) Learning locomotion over rough terrain using terrain templates. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Koh K, Kim S, Boyd S and Lin Y (2007) An interior-point method for large-scale L1-regularized logistic regression. *Journal of Machine Learning Research*. Koh et al (2007): Vol 8, pp 1519–1555

Kohl N and Stone P (2004) Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2619–2624.

Kolter et al. (2007): Kolter JZ, Abbeel P and Ng AY (2007) Hierarchical apprentice- ship learning, with application to quadruped locomotion. In Advances in Neural Information Processing Systems 20, pages 769–776. MIT Press, Cambridge, MA, 2008

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

Likhachev M, Gordon G and Thrun S (2003) ARA*: anytime a* with provable bounds on suboptimality. *Advances in Neural Information Processing Systems*.

McGhee RB (1967) Finite state control of quadruped locomotion. *SIMULATION* 9: 135–140.

McGhee RB and Frank AA (1968) On the stability properties of quadruped creeping gaits. *Mathematical Biosciences* 3: 331–351.

Mistry M (2009) *The Representation, Learning, and Control of Dexterous Motor Skills in Humans and Humanoid Robots*. Ph.D. thesis, University of Southern California.

Mistry M, Buchli J and Schaal S (2010) Inverse dynamics control of floating base systems using orthogonal decomposition. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*.

Mistry M, Nakanishi J, Cheng G and Schaal S (2008) Inverse kinematics with floating base and constraints for full body humanoid robot control. In *Humanoids 2008. 8th IEEE-RAS International Conference on Humanoid Robots, 2008*, pp. 22–27.

Murao H, Tamaki H and Kitamura S (2001) Walking pattern acquisition for quadruped robot by using modular reinforcement learning. In *Proceedings IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3, pp. 1402–1405.

Nakanishi J, Cory R, Mistry M, Peters J and Schaal S (2008) Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research* 27: 737–757.

Ng AY (2004) Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning* ACM, Banff, Alberta, Canada. New York: ACM Press, 78.

Raibert M (1986) *Legged Robots that Balance*. Cambridge, MA: MIT Press.

Raibert M, Blankespoor K, Nelson G and Playter R (2008) BigDog, the rough-terrain quaduped robot. *Proceedings of the 17th International Federation of Automation Control*, April 2008, pp. 10822–10825.

Ratliff N, Zucker M, Bagnell JA and Srinivasa S (2009) CHOMP: gradient optimization techniques for efficient motion planning. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*.

Rebula JR, Neuhaus PD, Bonnlander BV, Johnson MJ and Pratt JE (2007) A controller for the LittleDog quadruped walking on rough terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Saranli U, Buehler M and Koditschek D (2001) Rhex: A simple and highly mobile hexapod robot. *The International Journal of Robotics Research* 20: 616.

Schaal S (2009). *The SL Simulation and Real-time Control Software Package*. Technical report, University of Southern California. http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf.

Scholkopf B and Smola A (2003) A short introduction to learning with kernels. In *Advanced Lectures on Machine Learning*, pp. 41–64.

Sciavicco L and Siciliano B (2000) *Modelling and Control of Robot Manipulators*. Berlin: Springer.

Sentis L (2007) *Synthesis and Control of Whole-body Behaviors in Humanoid Systems*. Ph.D. thesis, Stanford University.

Smith J and Poulakakis I (2004) Rotary gallop in the untethered quadrupedal robot Scout II. In *Proceedings 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004 (IROS 2004)*, Vol. 3.

Sniedovich M (2006) Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and Cybernetics* 35: 599.

Stonier D and Kim J-H (2006) ZMP analysis for realisation of humanoid motion on complex topologies. In *IEEE International Conference on Systems, Man and Cybernetics, 2006 (SMC'06)*, Vol. 1, pp. 247–252.

Sugihara T, Nakamura Y and Inoue H (2002) Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control. In *Proceedings IEEE International Conference on Robotics and Automation, 2002 (ICRA'02)*, Vol. 2.

Ting JA, D'Souza A and Schaal S (2007) Automatic outlier detection: A Bayesian approach. In *IEEE International Conference on Robotics and Automation*.

Vukobratovic M and Borovac B (2004) Zero-moment point: thirty five years of its life. *International Journal of Humanoid Robotics* 1: 157–173.

Zucker M, Bagnell JAD, Atkeson C and Kuffner J (2010) An optimization approach to rough terrain locomotion. In *IEEE Conference on Robotics and Automation*.

Appendix: Index to Multimedia Extensions

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

Table of Multimedia Extensions

Extension	Type	Description
1	Video	Narrated video showing LittleDog traversing different kinds of terrains