The International Journal of
Robotics Research
30(2) 216–235
©The Author(s) 2011
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364910390538
ijr.sagepub.com

Comprehensive summary of the Institute for Human and Machine Cognition's experience with LittleDog

Peter D Neuhaus, Jerry E Pratt and Matthew J Johnson

Abstract

We discuss the main issues and challenges with quadrupedal locomotion over rough terrain in the context of the Defense Advanced Research Projects Agency's Learning Locomotion program. We present our controller for the LittleDog platform, which allows for continuous transition between a static crawl gait and a dynamic trot gait depending on the roughness of the terrain. We provide detailed descriptions for some of our key algorithm components, such as a fast footstep planner for rough terrain, a body pose finder for a given support polygon, and a new type of parameterized gait. We present the results of our algorithm, which proved successful in the program, crossing all 10 terrain boards on the final test at an average speed of 11.2 cm/s. We conclude with a discussion on the applicability of this work for platforms other than LittleDog and in environments other than the Learning Locomotion designed tests.

Keywords

Cognitive robotics, control architectures and programming, design and control, legged robots, mechanics, motion control, quadruped robot

1. Introduction

Legged Locomotion holds the promise of allowing robots to navigate rough terrain not accessible by wheeled and tracked robots. To that end, the Defense Advanced Research Projects Agency (DARPA) conducted a three-year program entitled Learning Locomotion (L2) to investigate quadrupedal locomotion over rough terrain. We were one of six university teams to compete in the program with the goal of achieving the fastest overall speed across a variety of terrains. In this paper we explain our approach, summarize some lessons learned from the project and provide some insight into the aspects of the program that we feel can be applied beyond the L2 program.

The L2 program was, at its core, a quadrupedal locomotion competition. To allow the teams to specifically focus on locomotion, the issues of hardware, sensing and localization needed to be mitigated. The L2 approach was to provide each team with the same equipment and make the competition focus on control algorithm development.

The hardware provided was the Boston Dynamics LittleDog (Figure 1), which has 12 actuated degrees of freedom (DOFs) with high gear reduction servo motors powering each joint. Each leg has three actuated DOFs: two at the hip, and one at the knee. The two motors at the hip actuate the hip pitch and roll DOFs using a differential. Sensing is addressed by providing a high-resolution (1 mm) height map file generated by a laser scan for each terrain board.

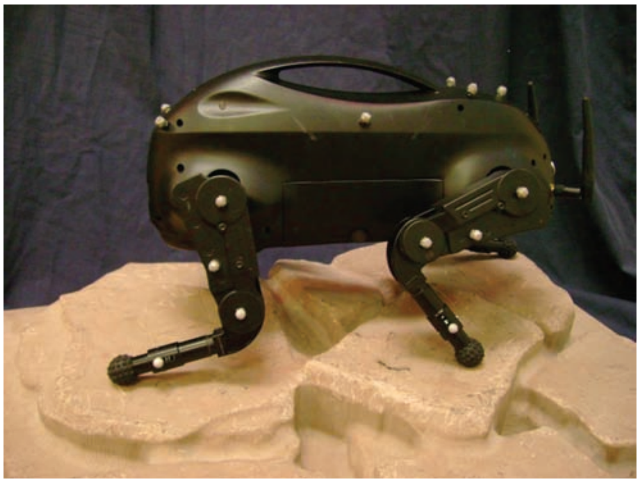


Fig. 1. LittleDog robot provided by Boston Dynamics, Inc.

A Vicon motion capture system (mocap), comprised of six cameras located above the experimental area, provides very accurate localization of both the terrain and the robot. In

Florida Institute for Human and Machine Cognition, USA

Corresponding author:
Matthew Johnson
Florida Institute for Human and Machine Cognition, 40 South Alcaniz Street, Pensacola, FL 32502, USA.
Email: mjohnson@ihmc.us

addition, the LittleDog also has an inertial measurement unit (IMU), which provided redundant body orientation information, and a three-axis force sensor in each foot.

The LittleDog robot has an internal computer controller that runs a feedback loop at 500 Hz. This loop, developed by Boston Dynamics, is responsible for performing the PD position control on the 12 joints. The LittleDog robot communicates at 100 Hz with an off-board computer. The off-board computer is a desktop computer that runs the team's control algorithm. The off-board computer sends the robot the desired joint angles and velocities, the desired feed forward values, and the proportional–derivative (PD) gains. The robot sends back to the off-board computer the IMU values, applied current to the motors, and foot force values.

The teams were evaluated monthly at the DARPA test site, which had duplicates of the terrain boards, as well as unknown terrain boards described as 'of the same class.' Some of the classes tested were rocks, logs, slopes, steps, barriers, gaps, wavy dunes, and a peg board (stepping stones). Obstacles were typically large relative to leg length, with some obstacles 65% of the leg length. This measure does not accurately describe the full problem, because LittleDog has a portion of the body that is below the hip joint. If you consider the leg height as the height of the lowest portion of the body, the obstacles were approximately 83% of the leg height. Over the course of the project 28 different terrain boards were crossed, making this the most extensive and diverse test of quadrupedal locomotion over rough terrain to date. The task was to successfully navigate across the terrain challenge to reach an arbitrarily specified goal. The teams were given three attempts to cross each board and were evaluated by the best average speed of their top two runs. Any run that did not reach the goal received a speed score of zero, so although speed was the main metric, both reliability and robustness were important in ensuring a good score.

We will begin our paper with a brief review of related work. Next we explain what we felt were the major challenges and issues associated with the L2 project. We then present our approach in general followed by a section that highlights the details of a few key aspects of our approach. Next we discuss some tools and techniques that we developed that were found to be helpful during the project. We then summarize our results and the major lessons learned over the course of the project. Finally, we discuss how this program applies to future applications beyond LittleDog and L2.

2. Related work

Most quadrupedal legged locomotion work prior to L2 focused on flat terrain. The focus on rough terrain distinguishes this work. Due to the difficulties involved with legged robots, quadrupedal locomotion on rough terrain has mostly been studied in simulation. For example, the task of searching through known terrain with impassable zones lends itself to formulation as a search problem, so this problem is often treated only in simulation with simplified models, as in Pack and Kang (1999). Oomichi et al. (1994) have worked on determining suitability of terrain for legged locomotion as part of a hierarchical controller; their path and gait planning algorithm validation was again performed in simulation.

Another approach avoids the issue of sensing the environment and path planning by writing controllers that adapt in real time to the given terrain. Prajoux and Martins (1996) discussed a controller that adapts to foot slippage using force control in simulation. Estremera and Santos (2002) approached quadrupedal rough terrain navigation by combining gaits that maintain constant body movement with gaits that break the movement phases into separate body movement and leg movement phases. The algorithm was validated using a rotationally symmetric robot on level terrain with software-defined 'forbidden zones' and the hybrid gait was derived assuming pitch and roll remained close to zero. Our robot is asymmetric and overcoming some types of terrain requires significant pitch and roll, so the algorithm was not directly applicable. Their more recent work (Estremera and Santos, 2005) presents a gait that does function on modestly uneven terrain (6.5 cm step compared to 48 cm extended leg length, equating to 13% of leg heights), but the terrain is composed of simple steps.

Because of the accurate terrain model and contest arrangement of the trials, the problems and issues presented to the L2 teams represented a unique set of challenges. The work that was most relevant as a basis for our work and as a point for comparison was the work of the other L2 teams.

Given a high-resolution terrain height map, each team took a different approach to deciding where to step. Work by the University of Southern California (USC) team (Kalakrishnan et al., 2009) used terrain templates to rate the quality of the footholds. The technique looks at the terrain height around the point of interest at different scales in order to search for various features. This technique offers the benefit that the ranking function can be learned through various approaches. The Stanford team (Kolter et al., 2007) used demonstration by an expert to train the reward function for feature weights.

As most teams determined, the path for the center of the body should pass through the double support triangle (referred to as the common triangle in this paper). USC (Buchli et al., 2009) further went on to reduce the problem to connecting points on the diagonal pair lines (which we refer to as trot lines). They also recognized that the sway of the body should be controlled as a function of speed and stability.

One of the first teams to develop highly dynamic maneuvers for the LittleDog was the Massachusetts Institute of Technology (MIT) team (Byl et al., 2008). After that, most teams developed their own form of dynamic jump and lunge. Although each had its own implementation, they all allowed the robot to cross the obstacle.

Some work has been done on developing quadrupedal gaits that can be parameterized for smooth transition from crawl to trot. The use of the gait duty factor (Hirose et al., 2009) allows for dynamic and static fusion gait. This parameterization, however, only works when swing times for all legs are equal.

3. Issues and challenges

While there were many difficulties inherent to the project, namely the aggressive testing schedule, complex and diverse terrain types, and use of unknown terrain boards, these were not the primary limiting factors. We now discuss our interpretation of the major challenges of the L2 project.

3.1. Terrain scoring and planning

Generic terrain scoring was a challenge. Initially we tried applying machine learning techniques to the problem, but it was frequently difficult to automatically determine the cause of failure in order to get the feedback needed for a machine learning algorithm. For example, if a robot falls during a step, it is difficult to determine if the cause was due to something wrong with the swinging leg, or a slip on one of the stance legs. Heuristic-based approaches were also challenging because of the large variety of terrain types that needed to be address for this program

Quadrupedal locomotion planning had numerous challenges, but the two main ones were bounding the search space and guiding the search, both necessary to find a solution in the limited time allotted. Given a 12 DOFs robot and a high-resolution terrain model, it is essential to reduce the search space in order for the problem to be solvable in the allotted time. Guiding the search was also critical. In order to apply algorithms, such as Dijkstra's algorithm, one needs a good cost function as well as an expected cost-to-goal function. All of the obvious cost functions seem to have limitations. For example, distance to goal causes extra, unnecessary searching when the robot has to go around an object and when using the minimum number of steps it is hard to determine an estimated remaining cost to goal. The main difficulty in a search algorithm is that each step limits the kinematic reachability and these cost functions do not account for the coupling between legs and diagonally opposite steps inherent in a quadruped.

Another major issue that we dealt with was the interdependency between terrain scoring, planning, safety factors for successful execution, and the execution itself. Assigning 'blame' to a particular component of the algorithm when the run was unsuccessful or slow was a difficult task. A slip of the robot off an edge resulted in the following questions: Was it because the terrain scoring allowed points too close to the edge? Was it because the swing leg tracking was poor? What is because the planner should have found a better plan?

3.2. Body geometry

Some issues were directly related to the specific hardware configuration we were using. One issue was that as the leg swings forward the knee has to pass below the bottom of the body. This led to complex collision avoidance algorithms to avoid knee collision while swinging.

Other issues were related to hardware limitations. The low bandwidth (100 Hz) loop that was available to the higher level controller off-board computer limited the potential for very reactive feedback control algorithms. In addition, LittleDog is a typical position control robot and could not implement full compliant control algorithms. It did have foot force sensors, but we deemed them insufficient for effective force control at high motor speeds due to the update rate of the off-board controller.

3.3. Foot slippage and body collisions

A key execution issue was handling large, unplanned, body movements due to foot slippage and body collisions. Foot slippage is one of the main challenges of the rough terrain environment. Small foot slips can lead to a foot falling off of an edge, resulting in unexpected stances and a possible loss of stability. In addition, some of the terrains required risky footholds to be used. Slow and cautious crawl gaits can limit the impact of foot slippage, but the L2 focus on speed meant foot slippage was a primary factor. Body collisions, including shin and knee, were also a challenge. Rough terrain can obstruct a leg's path while swinging from one location to another. In addition, even with fixed foot positions, shifting the body center of mass changes the shank link angle relative to the terrain, which can result in a leg segment contacting the terrain accidentally. Anticipating, detecting, and avoiding or dealing with such collisions were important challenges during the project.

3.4. Swing speed

The last major issue, and probably the most important during the final phase where obtaining higher speeds became critical, was the slow swing speed combined with the low center of mass height of the LittleDog platform. This combination made reactively responding to slipping and tipping very difficult, as well as dynamic maneuvers such as jumping, trotting, and bounding.

To compare LittleDog to other robots and animals, we can look at a non-dimensionalized dynamic parameter:

$$P = \frac{\sqrt{\frac{z_{\text{com}}}{g}}}{T_{\text{swing}}}$$

where g is the gravitational constant, z_{com} is the height of the center of mass, and T_{swing} is the leg swing time. This dynamic parameter is essentially a legged robot's or animal's pendulum falling time constant divided by its swing time. For a human, z_{com} is approximately 1.0 meter and T_{swing} is approximately 0.3 seconds, giving a dynamic parameter of approximately one. BigDog (http://bostondynamics.com/robot_bigdog.html), a dynamic robot that is impressive at push recovery, has a similar value, perhaps slightly higher due to its incredibly quick legs. LittleDog, on the other hand, has a center of mass height of the order of 15 cm and a leg swing time of around 0.7 seconds for a typical swing. This results in a dynamic parameter of approximately 0.2, significantly less than a human or BigDog. In other words, LittleDog falls quickly and swings slowly, limiting its reflexes when disturbed and limiting its dynamic maneuverability.

In addition, since LittleDog's swing speed is slow, the swing leg must be taken into consideration when determining body motions in order to ensure that the leg can swing in time for a given body motion. In contrast, for a robot with fast swing speed, the body motion can be determined with little regard to the swing legs, knowing that the swing legs can be controlled to keep up with the body, even for quick body motions. Due to this coupling between swing and body shift on LittleDog, we often had to use iteration to simultaneously solve for body and swing motions.

4. The approach

There were five other teams competing in the L2 program, each using the same equipment and working on the same terrain. Due to the obvious high-level decomposition of the problem of getting across the terrain as fast as possible, there was a lot of consistency in the approaches across teams when viewed from a high level. The problem required evaluation of the terrain, planning, and then execution, which typically included a layer of reactive behaviors. In this section we will highlight some of the distinguishing features of our approach.

Each team had three trials to cross the terrain. For each trial, the robot was placed on the terrain and the teams were given 90 seconds to 'think' about how to get to the goal. We used this 'thinking' time to generate multiple footstep plans and evaluate each to find the best one. Once the run began, we would generate a stance list for each successive pair of steps. At each 10 ms control cycle we would get the next stance from the list, adjust it with reactive control, and determine the joint angles for the 12 DOFs. The reactive modules resided inside this loop to handle modeling and tracking errors, as well as unexpected situations. This included an option to discard the current plan and return to the planning stage if things were significantly off plan. The overall architecture is shown in Figure 2.

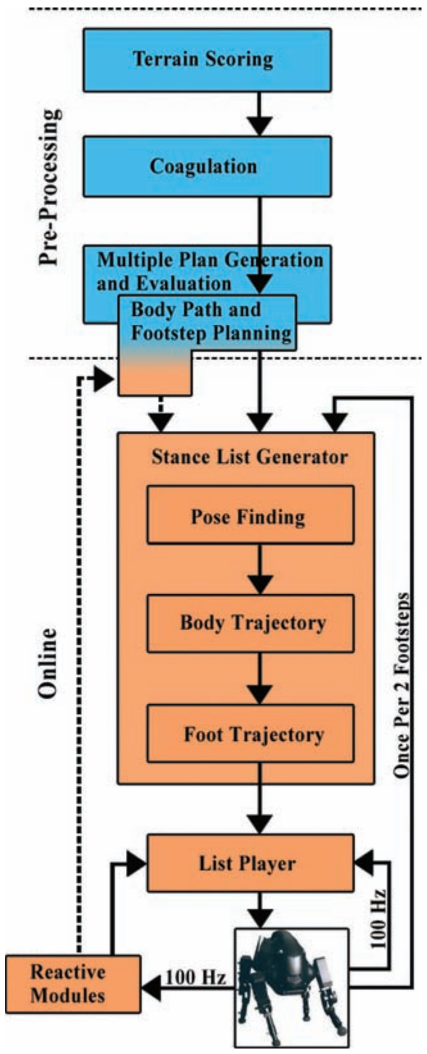


Fig. 2. Functional diagram of our approach. Figure adapted from a similar diagram presented by the USC team.

4.1. *Terrain scoring and coagulation*

The terrain model was provided as a high-accuracy millimeter resolution height map. At the time of scoring the terrain, we did not know the start location of the robot. However, we knew the goal location and the terrain file was a 0.6 m × 1.8 m rectangle, so we could deduce the approximate direction of robot travel.

4.1.1. Compensating for body geometry We scored the terrain on a millimeter scale grid. Because of the leg geometry, the knees bend toward the center of the body. For the front legs, the knees face backward, like a bird's ankle, and for the hind legs, the knees face forward. This leg arrangement means that a front leg could step at the base of a step up without colliding with the step, while the hind legs could step close to the base of a step down (see Figure 3). Thus, each terrain point was assigned a front leg score and a hind leg score. The score was scaled between 0.0 and 1.0, and then grouped into one of four categories, Great, Good, Risky, or Bad, by value.

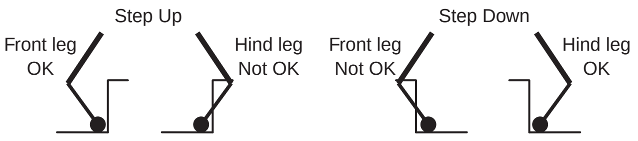


Fig. 3. On the left is a step up. Because front legs bend backward, they can get close to the base of a step up. Because hind legs bend forward, they can step close to the base of a step down.

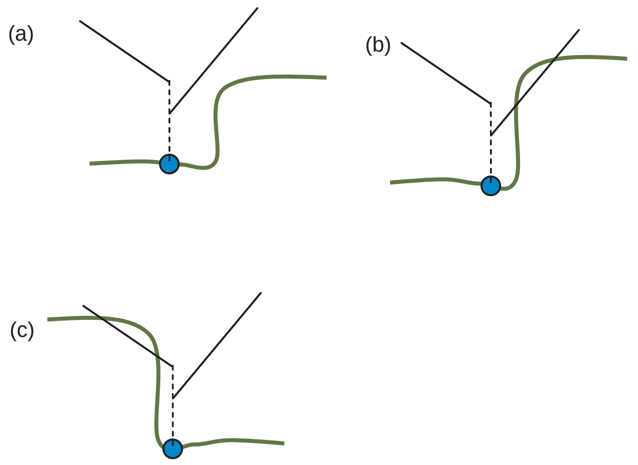


Fig. 4. The Vee terrain scorer was used to make sure that the terrain in front and behind a potential foothold point in question was clear. The three figures show the terrain (green line), the potential foothold point to be scored (blue ball), and a cross section of the Vee scorer. For a front leg, the front of the Vee is used to keep the swing trajectory clear and the back of the Vee is used to avoid points in which the shin might hit the terrain while the foot is on the ground. If any part of the terrain is above the black lines of the Vee scorer, then the foothold point gets marked as a Bad point. The Vee scorer is shown for a front leg traveling to the right. Figure (a) shows a good foothold point. Figures (b) and (c) show bad foothold points. (Color online only.)

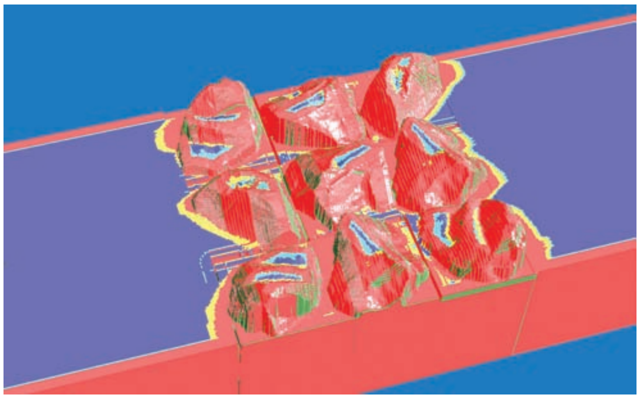


Fig. 5. The terrain score for a front leg, with the direction of travel from lower left to upper right, is overlaid on the terrain. Red indicates a Bad score, Yellow is Risky, cyan is Good, and blue is Great. (Color online only.)

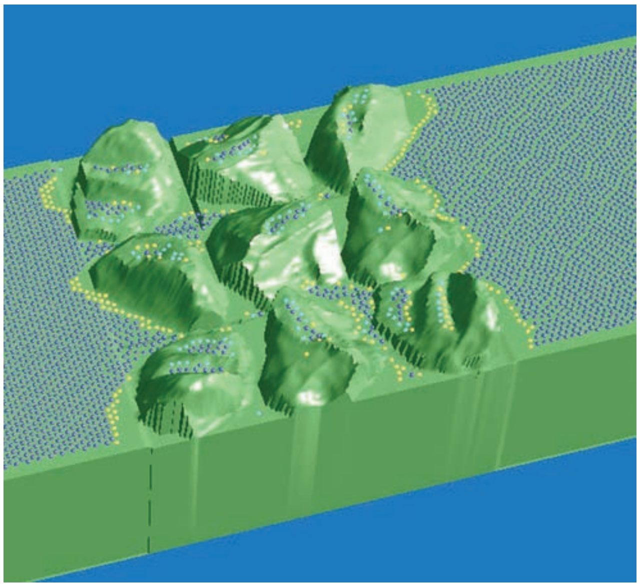


Fig. 6. The coagulated scored terrain points are shown overlaid on the terrain. The blue dots are Great points, the cyan dots are Good points, and the yellow dots are Risky points. (Color online only.)

To determine the score for a potential foothold point, we combined the score from a set of sub-scorers, with each sub-scorer evaluating the terrain for a given feature. One sub-scorer used the height of the terrain at a given radius from the point to detect drop offs and bases of cliffs. Another sub-scorer used the local surface normal. A third scorer verified that an asymmetrical Vee (see Figure 4) profile centered at the potential foothold point and aligned with the direction of travel does not intersect the terrain. The Vee scorer was used to avoid placing the foot where there was higher terrain in front or behind the foothold point. For front legs, terrain behind the foothold point could cause problems by hitting the shins. Terrain in front of the foothold point could cause problems when the leg tried to swing. The parameters for the scorer were tuned based on the type of terrain. A scored terrain is shown in Figure 5.

In order to reduce the number of terrain points for the planner to consider, the scored terrain points were *coagulated* (see Figure 6). This coagulation process sorted all the potential foothold points by score. Starting with the foothold point with the highest score, all the points in a circle around it were coagulated, or combined, into this point. The process was then repeated for all subsequent points, except for Bad foothold points. In order to preserve terrain edges and features, points were coagulated only if they were at similar heights.

4.2. *Two-stage path planner*

The role of the path planner was to produce a speed-optimized set of ordered footsteps, using the coagulated terrain points, which start with the initial stance and end at the goal point. In order to help the planner determine in

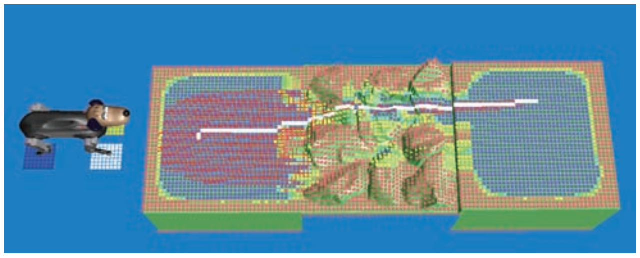


Fig. 7. The planner approximate body path is shown as the white line. The red, blue, and yellow reflect the values from the cost function used for the Dijkstra's search algorithm. The patches under each of the dog's feet present the area used for evaluating the terrain at each body position. (Color online only.)

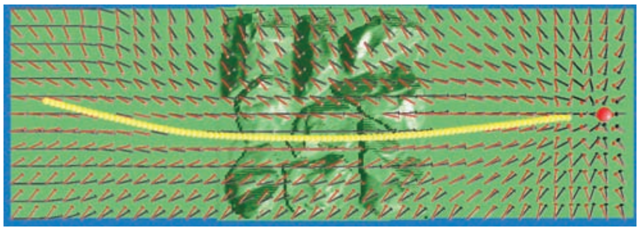


Fig. 8. The approximate body path is shown by the yellow line. The vector field of red arrows shows the step direction for a given body center. The vector field of gray arrows shows the suggested body yaw for stance finding. (Color online only.)

which direction to search for the next footstep, an approximate body path (see Figure 7) was generated for each plan. The approximate body path started at the body center start location and ended at the goal.

The approximate body path was generated using Dijkstra's search algorithm with the cost being related to the shape of the terrain and the distribution of coagulated terrain points. For each node (body center position) of the search graph the robot was placed at that location on the terrain in its nominal stance (see Figure 7), with foot pads presenting the most likely places that the foot would contact the ground. The height variation of the terrain within each foot pad, as well as the height variation between foot pads, was considered in determining the cost of each node. In addition, the distribution of the coagulated terrain points within each foot pad was considered. In order to avoid potential body collisions with the terrain, the height of the terrain at the body center was compared to the height of the terrain under each pad. If the terrain height at the body center was much higher than that of the foot pads, then the cost of that node was increased due to the increase likelihood that the body would collide with the terrain.

In order to generate varying plans for the same start position, the approximate body path was shifted slightly to the left or right. The approximate body path was used to generate a vector field (see Figure 8) for the step direction and suggested body yaw. These components were used to keep the body following the path and in line with the direction of travel, respectively.

Our planner took as input the scored terrain (coagulated points), the current stance, the goal point, and the approximate body path. We also enforced a regular gait (front left, hind right, front right, hind left) common to many crawl gaits found in nature, as well as being used by all other LittleDog teams. Given the start stance, we chose the appropriate swing leg and enumerated a subset of possible footsteps. These were ordered from best to worst based on an optimal step heuristic described in Section 5.2. We used a depth-first search over the ordered footsteps to plan a path to the goal. The heuristic proved good enough to allow depth first to provide very good solutions for all terrain boards

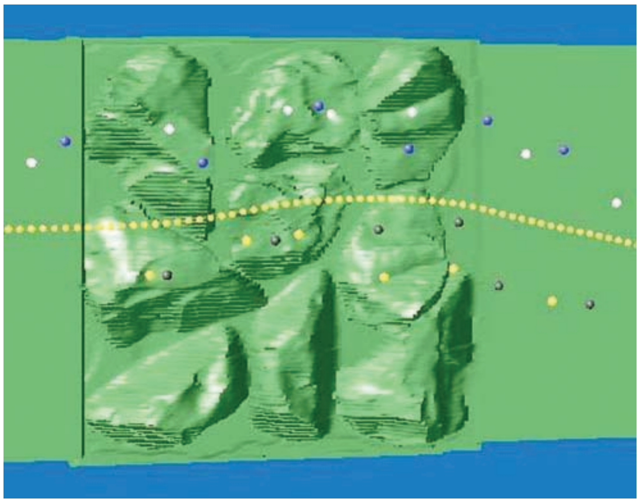


Fig. 9. The output of the planner was an approximate body path and a set of footsteps from the start stance to the goal point. The yellow series of dots show the approximate body path. The white dots show the front left footsteps, the yellow dots show the front right footsteps, the blue dots show the hind left footsteps, and the black dots show the hind right footsteps. (Color online only.)

we encountered. Using depth first also provided the planner with the computational speed we required in order to allow for real-time replanning in the event that the robot slipped or fell. As we expanded search tree nodes on our depth-first search, we performed numerous feasibility checks, such as kinematic reach, body and leg collisions, incircle radius of the support triangle, and static stability. We performed these checks on both the touchdown of the proposed swing and the projected liftoff of the following swing leg. We sequenced the checks from fastest to slowest to minimize planning time. The last check that was performed was a search for a valid pose using our Spring Based Pose Finder (described in Section 5.1) to ensure there exists at least one valid pose for the given feet positions. The result of this process was a recommended sequence of footsteps from the start position to the goal point (see Figure 9).

4.3. Execution

The execution of the trial was controlled by a multi-level state machine. The top level controlled the planner and gait type, which was selected from the following: trot, crawl, Xgait (see Section 5.5), lunge, and jump. The trot gait had two legs swinging at all times. This gait was very fast, but only worked on flat ground. The crawl gait had at most one leg swinging at a time, and generally had a portion of the time with no leg swinging. The crawl gait was primarily used for extremely rough terrain. The Xgait was used for terrain that was moderately rough. The lunge was used when the body center was close to the goal and the robot dove or lunged to the goal. The jump was used for getting over or across large obstacles. Each gait contained another state machine, called the gait controller, which was responsible for generating and modifying joint trajectories in real time. During a given trial, the top-level controller changed state only a few times, while the gait controller usually changed with every step.

5. Key algorithm modules

This section describes some of the key algorithm modules that we developed and used for this project. It is the details of these modules that primarily differentiated our approach with those of the other L2 teams.

5.1. Pose finder

Given a set of selectable constraints, the pose finder was used to determine a valid pose for the robot's body given the three-dimensional (3-D) location of n feet, where n was three or four. The pose of the robot was defined as the 3-D position (XYZ) and the 3-D orientation (yaw, pitch, and roll) of the center of the body. Constraints that could be selected in the pose finder include the following.

Kinematic reachability: each foot must be within a given percent of its own actual kinematic workspace. If the value is 100%, then the applied kinematic workspace matches the true workspace of the robot. Values less than 100% were used to allow a factor of safety, making the constraint harder to meet. Values greater than 100% relaxed the kinematic workspace, effectively tricking the pose finder into thinking that the robot was bigger that it really was. This technique was used when a solution must be found at the expense of slightly violating the kinematic constraint.

- The projection of the body center on the ground plane must be inside the ground plane polygon formed by the support feet and away from any edge by a distance of the given static stability margin.
- The body was constrained to the specified XY position.
- The maximum change in the yaw of the body from a given yaw was limited.
- The body could not collide with the terrain.
- The legs could not collide with the terrain

Note that any combination of the constraints could be used depending on the situation. For each constraint violation, we computed a corrective body action that could fix the constraint.

We developed what we called a Spring Based Pose Finder. This pose finder started with a seed body pose for the first iteration. Each constraint was modeled as a virtual spring with one end attached to the body center and the other end attached to an external point. The distance from the body center to the external point was a function of how much the constraint was violated and direction was such that body motion in that direction would eventually satisfy that constraint. The spring was able to exert forces and/or torques on the body based on the location of the external point. With each iteration of the pose finder, the body pose was moved in the net direction of the combined spring force and torque. After each iteration the constraints were evaluated and the virtual body forces and torques were computed. In practice, we limited the number of iterations to 50, after which the algorithm gave up and no solution was found. The Spring Based Pose Finder algorithm did not guarantee to find a solution nor did it guarantee the optimality of the solution. The performance of the algorithm was a function of the seed pose and corrective action taken in response to a violated constraint.

The Spring Based Pose Finder was evaluated by comparing its performance to a grid-based brute force search algorithm. While this grid-based algorithm was also not guaranteed to find a solution, our use of a small grid size sufficiently minimized this problem. In a test of 1000 random support polygons, the Spring Based Pose Finder found a valid pose for 80 of the 1000, and the grid-based method found a valid pose for 210. However, the ones that the Spring Based Pose Finder missed were generally the ones in which, for the given support polygon, the largest volume in six-dimensional (6-D) pose space consisting of valid poses was very small; in practice, this translated into a poor choice for the support polygon. Thus, while the Spring Based Pose Finder did not always find a valid pose when one was available, it found the most valuable subset of all possible poses and did not return poses that translated into a poor choice for the support polygon. In addition, it was significantly faster than a grid-based search.

5.2. Optimal step location and available footstep enumeration

Our initial approach to planning was a traditional A* search. We experimented with a few cost functions based on different ways to measure distance to the goal. This approach resulted in long planning times due to the large search space. In addition, the cost functions did not result in fast trial times. The main reason has to do with the coordination of leg motions in a quadruped. For example, stepping too far forward with a hind foot will cause great difficulty

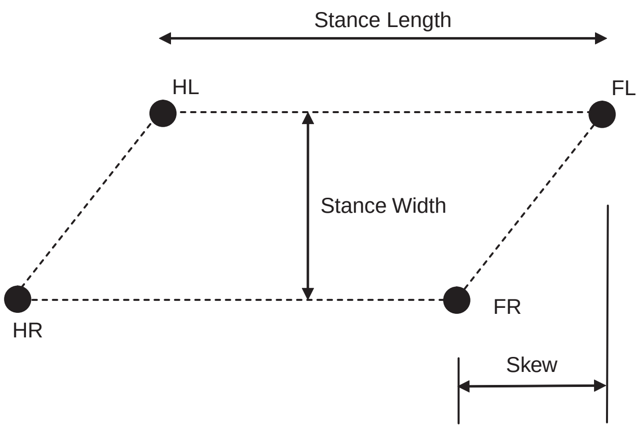


Fig. 10. This shows how the stance length, stance width, and skew are defined.

Table 1. Ideal stance parameters for flat ground walking.

Stance parameter	Value
Width	0.18 m
Length	0.30 m
Skew	0.09 m

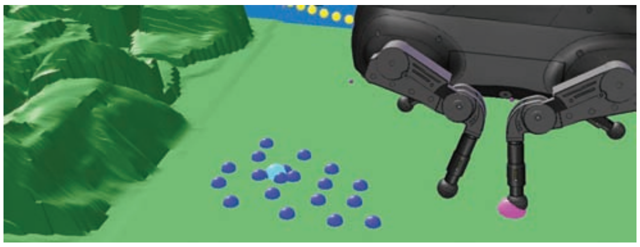


Fig. 11. The 'eyeball' shape of footstep enumeration. The optimal location is shown by the larger light blue ball. The enumerated steps are the smaller blue balls. Notice the cluster of footsteps around the optimal, the reduced density of footsteps once outside this core grouping, and the preference of wider over narrower. (Color online only.)

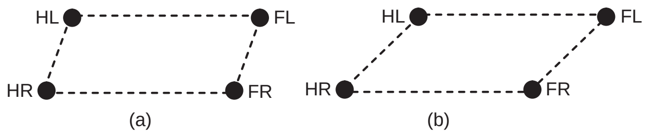


Fig. 12. The skew is a measure of how far ahead/behind the opposite foot is when the upcoming swing leg is a hind leg: (a) shows the feet with very little skew and (b) shows the feet with a significant amount of skew.

when attempting to shift the center of mass to facilitate lifting the opposite side hind foot two steps later.

Our final approach was to use an 'optimal' step location based on a fast walk on flat ground using a standard crawl gait. 'Optimal' refers to a location we determined through genetic algorithm tuning of gait parameters (stance length, width, and skew and body height) (Figure 10) in simulation, in conjunction with experiments with the LittleDog platform.

For flat ground walking we found the parameters in Table 1 to be good values. On any given step, however, the three support legs most likely did not line up with the ideal stance. Therefore, we chose the optimal step location as a linear combination of the three support legs in such a way that after four steps on flat ground the robot was guaranteed to be in a stance corresponding to the length, width, and skew parameters.

We then applied a weighted distance function to the available step locations. The weighting function accounted for three factors. The first was the distance from the optimal location. The second was the body relative direction. Taking a wider step was penalized more than taking a shorter or longer step. Taking a narrower step received an even larger penalty because of its impact on stability. The third factor accounted for in the weighting was the terrain score. Penalties were applied to step locations with lower terrain scores.

In order to reduce the search space and increase our planning speed, we enumerated our available footsteps using a coagulation technique similar to the one implemented by our ground scoring algorithm. We determined a feasible number of footsteps that can be evaluated at an acceptable speed (we typically used a branching factor of 20). Instead of taking the top scores, we selected the top 25% to get a cluster of the best points and then chose the next 75% using coagulation. During coagulation, after selecting a point based on weighted optimal score, we would remove points within a set distance. This provided a broader area of coverage within the allowable branching factor, as shown in Figure 11.

There were three main advantages to using the optimal step location as the heuristic to guide our planner's search. Firstly, the resultant plan tended to be very close to what we considered an optimal set of footsteps for a fast walk; deviations from optimal typically resulted in slower speeds. The second advantage was stability. Staying in a skewed stance (see Figure 12) of appropriate length and width was faster and more stable. Deviations from this optimal stance resulted in less stable stances and usually required more body movement, which often further destabilized the robot. The third advantage was planning time. The optimal step heuristic allowed us to use a depth-first search, resulting in a very fast plan time (of the order of 1–2 seconds for most terrains) and allowing us to replan in real time.

5.3. *Body path*

The path of the center of the body was determined in real time based on the planned set of footsteps. The body

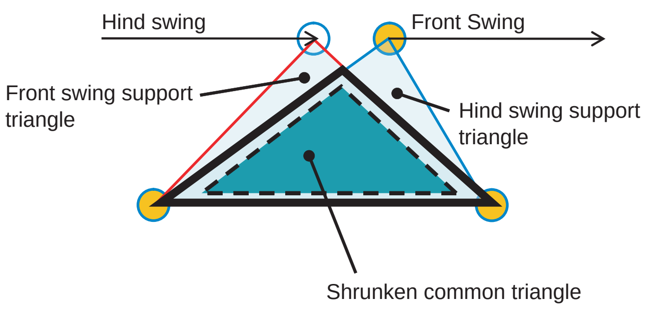


Fig. 13. The intersection between sequential support triangles, called the common triangle, in a same side hind-front swing leg exchange is shown by the thick black line. The common triangle is reduced by a safety factor, resulting in the shrunken common triangle (the dark green triangle). (Color online only.)

path was determined such that it satisfied the given stability requirements and minimized unnecessary body shifts. Because we always used a regular quadruped gait (front left, hind right, front right, hind left), every hind leg swing was followed by a same side front leg swing. A common triangle was formed by the intersection of the support triangle for the hind swing and the support triangle for the following same side front swing. To increase robustness, the common triangle was shrunken by a safety factor and the resulting triangle was called the shrunken common triangle (see Figure 13). A body shift between same side hind-to-front swing exchange was eliminated when the body shift for the hind swing ended inside the shrunken common triangle.

Figure 14 shows the body path for four consecutive steps, starting with the front left. The body path was formed by waypoints corresponding to the body center position at hind touchdown. For steps with no body shifts between same side leg swing transfers, hind leg touchdown was the same point as the front leg liftoff. These touchdown/liftoff waypoints were determined by placing a point on the single-support-leg-side median (see Figure 14) of the shrunken support triangle. The location along this median was governed by the parameter α, which was the percent along the median from the vertex. The parameter a was calculated for each hind touchdown (denoted by the numeric subscript) such that the angle between successive body path segments, θ, was optimized for a given terrain. The angle between successive body path segments controlled the body sway (Hirose et al., 2009). Intuitively, one might want to minimize θ such that the length of the body path is minimized. However, we found that the overall speed of travel was generally limited by the swing speed of the leg. Therefore, the body path was optimized for stability without affecting the speed by choosing a larger θ. Body sway helped increase stability by having momentum as the body center crossed the trot line. The trot line is the line connecting the two feet that stay fixed (i.e. do not swing) in a front and diagonal-opposite hind pair of steps. For example, for a front left and hind right step, the trot line is formed by the hind left and front right feet locations.

To minimize body accelerations, the body path was smoothed using a cubic spline (see Figure 15). For a given front and diagonal-opposite hind swing steps, the corresponding straight body path segment was smoothed by forming a spline with the current body position, the upcoming hind touchdown point, and the next hind touchdown waypoint. The path was only used for the given front and hind steps, up to the next waypoint. After these two steps a new splined path was generated based on the next four steps, or two waypoints.

5.4. *Convex hull and swing trajectory*

Swing trajectories were used to transfer the swing foot from its current position (start point) to the target position (end point). A valid and optimal trajectory was the one that did not collide with the terrain yet minimized the path length.

The trajectory was first formed by generating a two-dimensional (2-D) convex hull of the terrain by sampling the ground points between the start to the end point. In order to add a ground clearance safety factor, a circle was added to each sampled ground point before computing the convex hull. The radius of each circle ramped up from zero at the start point to the desired ground clearance after a given distance. Likewise, it ramped down at the end point. Because of the offset, the start and end of the trajectory was typically a sloped vector. This line we refer to as the lead out vector (out of the start point) for the start of the trajectory and the lead in vector (into the end point) for the end of the trajectory.

On very rough terrain, sometimes the best path for the swing trajectory was not a straight line, as seen from above, from the start to the end. This was the case, for example, if there was a tall rock (see Figure 16) between the start and end, and routing the swing trajectory to either side of the tall obstacle resulted in a safer and shorter swing.

To determine whether the trajectory should be adjusted from the straight path, the terrain height was examined along three parallel lines in the ground plane. The first one was the start to end line (the blue line in Figure 16). The other two lines (the red lines in Figure 16) were parallel to the first one, but offset on either side. The maximum terrain height was determined along each of the lines. If there was a feature along the center line that was a lot higher than either the start or end height, then the two offset lines were checked. If either line offered a lower maximum height, the lead out vector was tilted in the direction of the line that had the lower height.

5.5. *Xgait*

A common way to parameterize crawl gaits is with the duty factor, β (McGhee and Frank, 1968; Song and Waldron

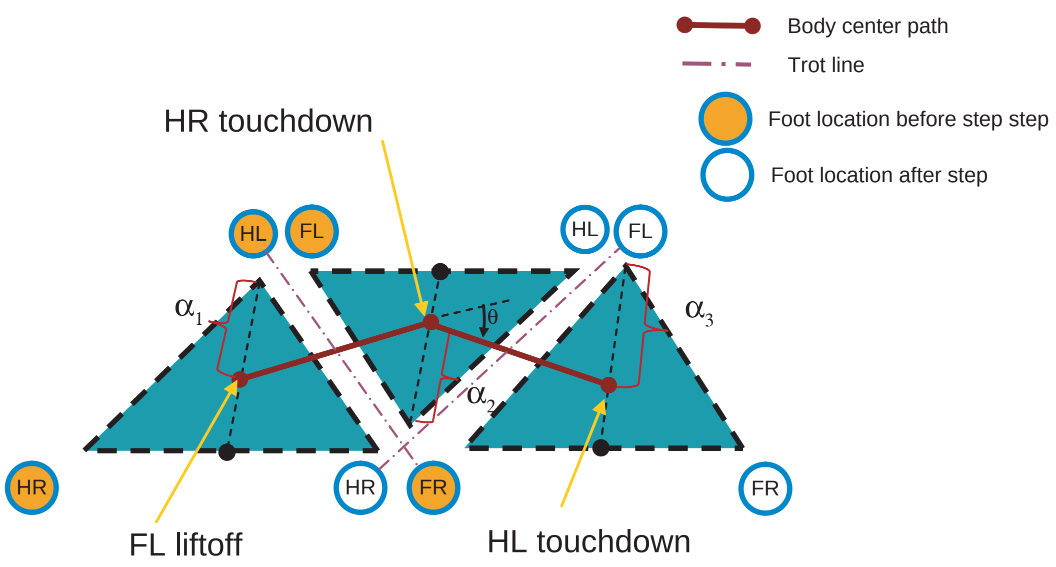


Fig. 14. The body path is shown for four leg swings: front left (FL), hind right (HR), front right (FR), and hind left (HL). The first straight section of the body path is for the FL and HR steps, and the second straight section is for the FR and HL steps. Successive hind touchdowns (note that a hind touchdown is the same point as the same side front liftoff) form the waypoints of the body path. These waypoints lie along the single-support-leg-side median of the shrunken support triangle.

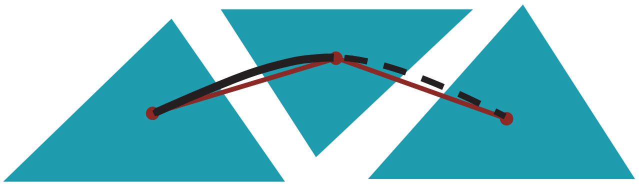


Fig. 15. To minimize accelerations the body path was smoothed using a cubic spline. For given front and diagonal-opposite hind swing steps, the corresponding straight body path segment was smoothed by forming a spline with the current body position, the upcoming hind touchdown point, and the next hind touchdown waypoint. Only the solid part of the splined curve was used. On the next pair of steps, the body path curve is regenerated using the next four steps.

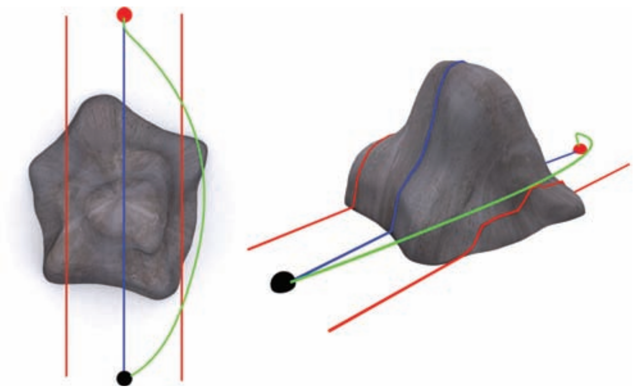


Fig. 16. A swing trajectory is needed to move the leg from the start position (the black dot) to the end position (red dot), while avoiding collisions with the terrain. We start with a line from start to end (blue line) and create offset lines on either side of this line (red lines). The maximum terrain height along each of the three lines is determined and the swing trajectory is shifted toward the lowest path. In this example, the path to the right avoids the large rock, so the swing trajectory (green line) is tilted to the right. (Color online only.)

1998). This parameter has been used by Hirose et al. (2009) to develop the dynamic and static fusion gait. Although this approach by Hirose et al. does allow for smooth transitions between crawl, dynamic crawl, and trot gaits, it has some limitations. The duty factor parameter only makes sense when describing an entire gait cycle, all four legs have a transfer or swing phase, and the duration of the transfer phase for each of the legs is equal. Another limitation is that the duty factor does not have a direct physical meaning with regard to the stability of the gait.

We developed the Xgait as a method of parameterizing a walking gait such that the same set of footsteps can be walked on as a static crawl, a trot, or anything in between. The use of the Xgait also allows smooth transition between a trot, static crawl, and dynamic crawl in real time.

The Xgait is based on the regular gait leg sequencing (front left, hind right, front right, hind left). It controls the timing of the leg swings and body shifts using two parameters. Each Xgait step (see Figure 17) consists of a front leg swing, the opposite diagonal leg swing, and a corresponding body shift.

The Xgait step is parameterized by two parameters; X_{FTD} (Front Touchdown) and X_{HLO} (Hind Liftoff) (see

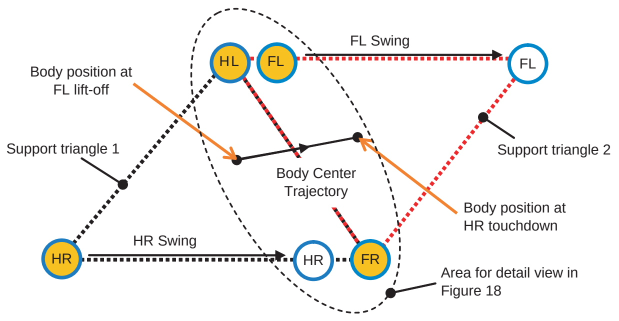

Body position at FL lift-off
Support triangle 1
HL
FL
FL Swing
FL
Support triangle 2
Body Center Trajectory
Body position at HR touchdown
HR Swing
HR
HR
FR
Area for detail view in Figure 18

Fig. 17. This figure shows the footsteps and body center shift for an Xgait step. In this step, the front left (FL) leg swings forward, the hind right (HR) leg swings forward, and the body shifts forward. Note that body center position starts inside support triangle 1. Also note that at the end of the Xgait step, the body center is in support triangle 2.

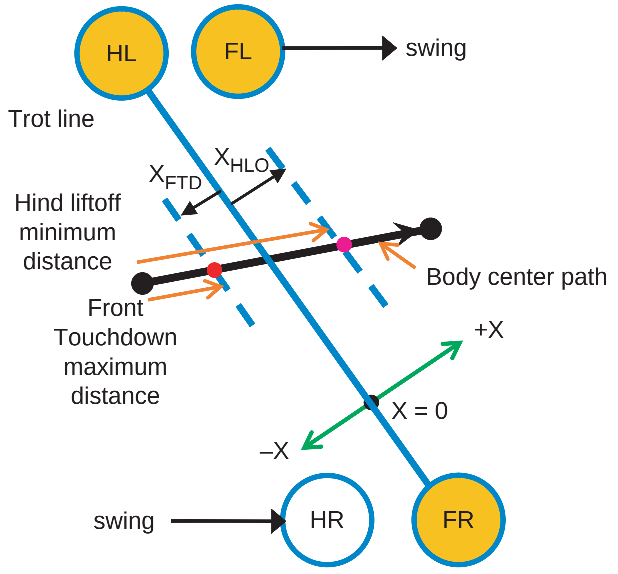


Fig. 18. This figure shows the parameterization for and Xgait step involving the front left and hind right swings. The figure is a detailed view from Figure 17. The view is from above with the foot positions projected onto a plane normal to gravity. The trot line is the line connecting the two support feet that do not swing during the Xgait step. The Xgait step is parameterized by the X_{FTD} and X_{HLO}. These are signed perpendicular distances to the trot line, with positive being in the direction of body center travel.

Figure 18). X_{FTD} is the signed perpendicular distance of the body center position to the trot line at the time the front leg touches down and ends it swing. From a stability point of view, if the body center position when the front leg touches down is *less* than X_{FTD}, then the step is more stable. Therefore, X_{FTD} is defined as a maximum, not to be exceeded, value.

X_{HLO} is the signed perpendicular distance of the body center position to the trot line at the time the hind leg lifts off the ground and starts swing. From a stability point of view, if the body center position when the hind leg lifts off is *greater* than X_{HLO}, then the step is more stable. Therefore, X_{HLO} is defined as a minimum value.

Physically, these distances are a measure of the static stability of the body. For example, if $X_{\mathrm{FTD}} < 0$, this means that the front leg touches down before the body center crosses the trot line (i.e. leaves the support polygon for the front swing leg). If $X_{\mathrm{HLO}} > 0$, this means that the hind leg lifts off after the body center crosses the trot line and enters the support polygon for the hind swing leg. The following conclusions can be made.

- If $X_{\mathrm{FTD}} \leq 0$ and $X_{\mathrm{HLO}} \geq 0 \rightarrow$ the Xgait is statically stable, with a quad shift.
- If $X_{\mathrm{FTD}} = X_{\mathrm{HLO}} \rightarrow$ hind leg liftoff occurs at exactly the same time as the front leg touchdown and there is no body motion with all four legs on the ground (no quad shift).
- If $X_{\mathrm{FTD}} \leq 0$ or $X_{\mathrm{HLO}} \geq 0 \rightarrow$ the Xgait is dynamic.
- If $X_{\mathrm{HLO}} \leq X_{\mathrm{FTD}}$ the hind leg will start swinging before the front leg touches down and the two swing trajectories will have some overlap in time. In this case, there is also no quad shift.
- If the X_{HLO} position coincides with the body center start and the X_{FTD} coincides with the body center end, then the Xgait is a full trot.

The Xgait approach determined the motion parameters and then controls the timing of the three motion events: the front leg swing, the hind leg swing, and the body center shift. The procedure for determining the Xgait is below.

Given:

- Four upcoming footsteps in a regular gait, starting with the front leg. The first two steps will be used in the upcoming Xgait move and the second set of two steps will be used to determine the end conditions for the first two steps.
- Body trajectory (x, y, z, roll, pitch, yaw) for associated with the four steps.
- Incoming body XY velocity.
- Detailed terrain information.

Determine for the next two steps:

- swing speed parameters (maximum velocity and acceleration) for two steps (front and hind) based on terrain;
- body XY end speed for the two steps;
- maximum body XY speed;
- maximum body XY acceleration;
- Xgait liftoff and touchdown parameters X_{FTD}, X_{HLO}.

While obeying the following constraints:

- the hind leg starts swinging at the same time or after the front leg;
- the swing times are a function solely of the Cartesian trajectory, not dependent on the trajectory of the body;

- the initial body speed matches the body end speed of the previous step;
- the final body speed is equal to or less than the target body speed;
- the actual body center position at front leg touchdown is less than X_{FTD};
- the actual body center position at hind leg liftoff is greater than X_{HLO}.

5.5.1. Swing speed move parameters The swing trajectory is a Cartesian curve in space that is determined based on the starting point, the step location point, and the shape of the terrain. The swing foot motion moves along that trajectory with a trapezoidal velocity profile.

During swing, the tracking of the desired swing leg position is inversely related to speed: the faster the swing, the worse the tracking. Therefore, the maximum velocity and acceleration of the swing leg is set based on the local features of the terrain. If the terrain is rough or a step error could be fatal (e.g. stepping off a cliff), the swing speeds are reduced. There are two swing speed levels, *Fast* and *Slow*. The *Fast* speed setting is used for smooth terrain with very little height variations, otherwise the *Slow* setting is used. For the *Fast* setting, the maximum speed is 100 cm/s and the acceleration is 20 cm/s. For the *Slow* setting, the maximum speed is 45 cm/s and the acceleration is the same, 20 cm/s.

5.5.2. Body XY move parameters For each type of terrain, we experimentally determined three different levels of body *XY* speed: *Slow*, *Medium*, and *Fast*. The numerical values for these three levels where tuned based on the motor limits and difficulty of the terrain type. The values used for three of the give terrain types are listed below.

During each step, the selection of speed level for the upcoming Xgait steps is determined as follows.

- If both front and hind steps start and end on flat ground, speed can be Fast
- The step direction, relative to the body heading, for the front and hind leg must be within 25 degrees of parallel for Fast speed setting, otherwise speed is either Medium or Slow.
- The Incircle radius of the support polygon for the hind leg swing must be larger than 5.2 cm for Fast speed setting, otherwise speed is either Medium or Slow.
- If one (front or hind) of the swing speeds is set to Slow, then the body speed is also set to Slow. This is because it does not help to have the body move quickly when the swing legs are moving slowly.

5.5.3. Desired end body XY speed The desired end body *XY* speed is based on the current step and the next two steps. The goal in setting the body *XY* desired end speed is to minimize the accelerations and decelerations of the body during each Xgait step and between Xgait steps. The second set of two steps are analyzed to determine what the best initial speed for those two steps are and the body *XY* end speed for the upcoming set of two steps is set to that value:

- get front swing move duration based on swing path and swing speed parameters;
- get hind swing move duration based on swing path and swing speed parameters;
- get splined body trajectory (see Figure 15);
- get maximum distance along the body trajectory – the body can be at the point of front leg touchdown (this is based on X_{FTD}, the location of the trot line, and the angle between the trotline and the body path);
- get minimum distance along the body trajectory – the body can be at the point of hind leg liftoff (this is based on X_{HLO}, the location of the trot line, and the angle between the trotline and the body path).

5.5.4. Synthesizing the two swings and body move into a single move The initial speed of the body needs to be checked to make sure that the front leg can start swinging immediately. If the initial body speed is too high, even with maximum deceleration, the body can potentially pass the maximum front touchdown point before the front leg can finish swinging. When the initial body speed is too high, the body speed is initially slowed and the front leg swing action is delayed.

The challenge in the Xgait is synthesizing the three motions (the two swings and the body) such that all the constraints are met. The solution can be arrived at iteratively. Figure 19 depicts the time constraints discussed next. For the first iteration, the body trajectory is determined using a minimum time motion (trapezoidal in velocity), which yields T_{B}, the move duration for the body. Given are T_{F} and T_{H}, the durations of the front and hind swing legs, which cannot be changed. Based on the determined body motion time profile, t_{F} is the time that the body center position is at X_{FTD}. If the front touchdown time occurs before the end of the front leg swing, the body motion must be delayed or slowed. The amount the body motion is delayed from the start of the swing is T_{wait}. If the initial speed of the body is non-zero, the body must start moving immediately, and T_{wait} must be zero. Otherwise

$$T_{\mathrm{wait}} = \mathrm{Max}(\,T_{\mathrm{F}} - t_{\mathrm{F}}, 0.0)$$

The time at which the body reaches the hind liftoff position is tH, which is the time that the hind leg starts swinging.

As shown in Figure 19(a), if the body motion ends before the hind swing ends (i.e. $t_{\mathrm{H}} + T_{\mathrm{H}} > T_{\mathrm{B}}$), then the body motion must be slowed down. This can be done by any combination of:

- reduce the maximum body velocity;
- reduce the maximum body acceleration;
- reduce the target body end speed.

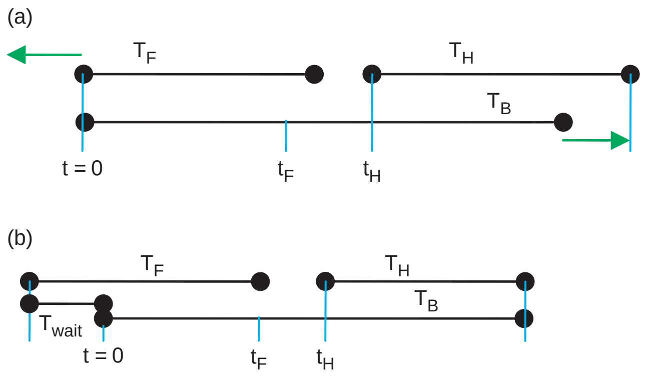


Fig. 19. This figure shows an example of the synchronizing of the leg swings and the body shift in the time domain. For the first iteration (a) in getting a synchronization solution, the body motion is started at the same time as the swing. In subsequent iterations in generating a solution (b), the start of the body motion was delayed with by amount T_{wait} so that the front leg touchdown occurred at the proper body position. The end of the body motion was synchronized with the hind leg touchdown by lowering the maximum allowable speed for the body.

In the Figure 19(b), the body motion has been slowed and the hind leg touchdown occurs simultaneously with the end of the body motion.

For each Xgait step, the governing parameters,X_{FTD} and X_{HLO}, are determined by the controller based on a number of factors, including initial body speed, quality of foot holds, spacing of the feet relative to each other, and smoothness of the terrain. For example, if the initial body speed at the start of a step is low, the X_{FTD} and X_{HLO} are selected for a static gait. If the initial body speed is high, the momentum of the body allows for a dynamic gait. Thus, these parameters can be selected for either a dynamic or a static gait.

In order to generalize this approach for other quadrupedal robot platforms, the X_{FTD} and X_{HLO} can be represented as dimensionless quantities related to the body size and leg length.

Figure 20 compares the Xgait for two different sets of Xgait parameters walking on the same set of planned footsteps, the only difference being the Xgait parameters. By changing the parameters from a static crawl gait (Figure 20(a)) to a dynamic crawl with overlapping swing legs, the quadruped is able to walk over two times faster.

5.6. *Dynamic jump maneuver*

A dynamic jump maneuver was used to cross terrain that was either impossible to cross in a statically stable way or faster to cross using a dynamic maneuver. The dynamic maneuver allowed the center of mass to get higher or farther than a static maneuver. The terrains that we used dynamic maneuvers on were the Gap, Steps, and Barrier.

The basic dynamic maneuver started with the four feet on the ground forming a rectangle. In order to unload the front legs, the center of mass was then moved to just a few centimeters in front of the hind legs. The body was then pitched up by extending the front legs while simultaneously applying torque to both hind leg hip pitch joints. During the move, the pendulum angle, the angle from the body center to the hind feet as projected on the sagittal plane, was calculated. When the pendulum angle went below a threshold level, the front legs started to retract and the hind legs started to extend, while still rotating in the pitch direction.

Once the center of mass was either on top of or over the obstacle, a series of obstacle-dependent open loop joint motions were executed. These motions brought the body into a standing position and ready to walk, jump again, or trot, depending on the upcoming terrain.

5.7. *Reactive modules*

During execution, the controller set the desired pose of the robot and position of the swing leg. There were numerous sources, including terrain modeling error, world position error, and loss of ground contact, which resulted in pose tracking error. To counteract these effects, we used real-time reactive modules that tried to minimize the pose and swing leg tracking error. The reactive modules included:

- touchdown detection;
- body pose and foot–ground contact;
- swing correction;
- tip detection;
- slip detection.

A few of the reactive modules required knowledge of whether the foot was in contact with the ground. Ground contact was determined using the three-axis foot force sensors. At start up, while the robot was off the ground, baseline levels for each of the three axes were determined. Ground contact was determined if the levels of any of the axes was above a threshold.

A touchdown detection module was used to stop the swing leg motion when ground contact was detected. The touchdown detection prevented accidental tipping when

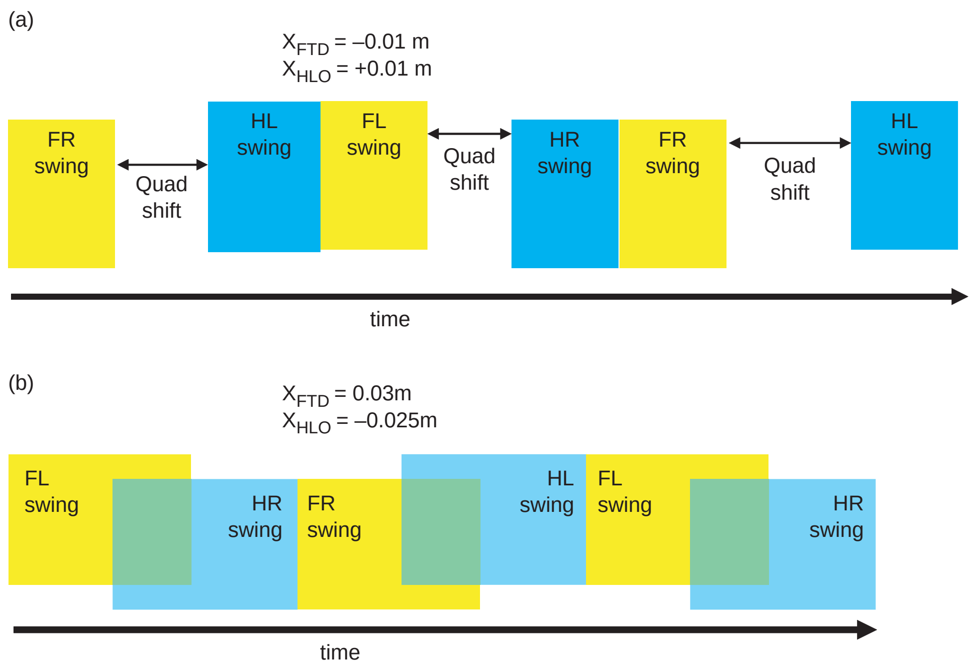


Fig. 20. This figure compares the Xgait for two different sets of Xgait parameters walking on the same set of planned footsteps, the only difference being the Xgait parameters. The yellow boxes represent when a front leg is swinging. The blue boxes represent when a hind leg is swinging. The green boxes represent an overlap between the front leg swinging and the hind leg swinging. In (a), $X_{\mathrm{HLO}} > X_{\mathrm{FTD}}$, therefore, there will be a quad shift. Note that there is no quad shift between same side hind to front leg transitions. This is generally true for optimized footstep plans. In (b), $X_{\mathrm{FTD}} > X_{\mathrm{HLO}}$, so there is an overlap in the swinging of the legs, as shown by the green boxes. The result is that the bottom set of parameters walked over two times faster than the top set of parameters. Note that the figures are not drawn to scale. (Color online only.)

the terrain was higher than expected and ensured ground contact if it was lower than expected.

The body pose reactive module controlled the position of the support feet relative to the body. This reactive control effort was implemented by adding offsets to the current desired feet positions. The pose tracking error was determined by combining real-time data from the mocap system and robot's IMU and comparing it to the desired pose. Support leg ground contact status was determined from the foot force sensors. Only support legs that were on the ground were used in the reactive pose control. If a support leg was determined to be off the ground, the leg was slowly extended down until ground contact was made. For feet that were in contact with the ground, their desired position was adjusted every controller cycle in order to correct for pose error.

It was important that the swing leg tracked its desired trajectory in the world, regardless of what the body was doing. Any error in foot placement could have resulted in stepping in a low terrain-score foot hold. For each control cycle, the controller used the stance from the list to determine the position of the swing foot relative to the body. Therefore, any error in the pose of the body during execution resulted in swing foot position error relative to the terrain. To account for this, the reactive swing foot correction controller adjusted the desired position of the foot to account for the pose tracking error.

One cause of failure was the robot tipping backward onto a hind leg that was swinging. We developed a reactive module that monitored the loading of the support legs in combination with the real-time pose of the robot to determine if the robot was tipping backward during the initial part of the hind swing phase. If the tip condition was detected, the leg was quickly put down, the swing and body shift were recalculated from the current position, and the swing phase was started over again.

In the event that the robot experienced a significant slip, the controller would immediately switch to an emergency recovery state. The emergency recovery state would have the robot try to stand up and start planning a new set of footsteps to the goal. Once the planner had generated at least four steps, the execution controller would start walking on the new footsteps while the planner simultaneously finished the plan.

6. Tools

During the course of this project, we developed many tools and techniques that enhanced our productivity.

6.1. *Frame geometry*

A common issue with multi-joint robots is reference frame management and transformation between reference frames.

The concept and process is well documented (Craig, 2004), but the practice can be tedious and error prone. One common error is to perform geometric operations on points and vectors that are in different reference frames. Another common error is to use the wrong transformation matrix when transforming from one frame to another. In particular, it is often confusing whether to use a given transformation matrix or its inverse.

Our solution for preventing reference frame problems was to develop Java classes to support reference frame geometry. We developed a class called *ReferenceFrame* that includes the transformation between itself and its parent frame. We developed classes called *FramePoint*, *FrameVector*, and *FrameOrientation* that have the same functionality as points, vectors, and orientations, respectively, but include the association with a reference frame. We now automatically verify reference frame matching prior to any operation involving two components. We are also able to freely transform from one reference frame to another with a single method call on a FramePoint or FrameVector. The internal software automatically determines the required ordering of transformation matrices. Although a very simple concept, this technique helped reduce the reference frame bugs that tend to plague robotic software.

6.2. Stance list

A stance was an object that represented the kinematic state (either desired or actual) of the robot. It contained the pose of the robot, which was the 3-D location of the body center as represented by a *FramePoint* and the 3-D orientation of the body. It also contained four footsteps, one for each foot. The footstep could be one of two types. The Cartesian footstep contained a *FramePoint* for the 3-D location of the foot. The other type was the joint position footstep, which specified the three joint angles of the leg, instead of the location of the foot. When constructing a stance, it was required that the reference frame of the body center position be the same as the reference frame of the four footsteps. The stance also had a list of the legs that were denoted as swing legs. This list could contain between zero and two legs.

From a desired stance, the desired joint angles for the 12 DOFs could be calculated. This was done by first calculating the vector from the body center to each foot and then using the inverse kinematics to calculate the joint angles.

A stance list was used to represent a sequence of stances in time. Each item in the list was separated in time by the controller period, which was 10 ms. During execution, a list of stances was calculated for each state. The list represented the body and swing leg motion for the upcoming state. During execution, any given stance list was typically 40–150 elements long, which corresponded to 0.4 seconds and 1.5 second for execution time. The motivation behind using the stance list was that body and joint motions could be planned ahead and the trajectories generated in a smooth way.

6.3. Non-dynamic simulator

We developed a non-dynamic simulator in order to quickly evaluate a plan. Using the starting stance from the plan and the planned footsteps to the goal, the non-dynamic simulator calculated the total execution time to get from start to goal. The non-dynamic simulator used the same execution controller, but the reactive controllers were disabled. As with the real-time execution, upon entering each controller state, the stance list for that state was generated. The length of the stance list represented the time that the controller would stay in that state. However, instead of using the stances in the list to set desired joint angles every 10 ms, perfect tracking was assumed and the last stance in the list was used to set the position of the robot for the next state. In this way, a combined list was generated that represented the leg and body motions, at 10 ms intervals, to get the robot from the start to the goal. The length of the list represented the ideal time the execution would be expected to take. Typically, the non-dynamic and the dynamic simulation were within 10% of each other, and the dynamic simulation was within 15% of the actual trial.

The non-dynamic simulation took about three seconds to compute. We initially used the non-dynamic simulation for debugging the swing trajectory and body path generation, because those components are mainly geometrical. Eventually, the non-dynamic simulator was also used to compare plans for a given trial.

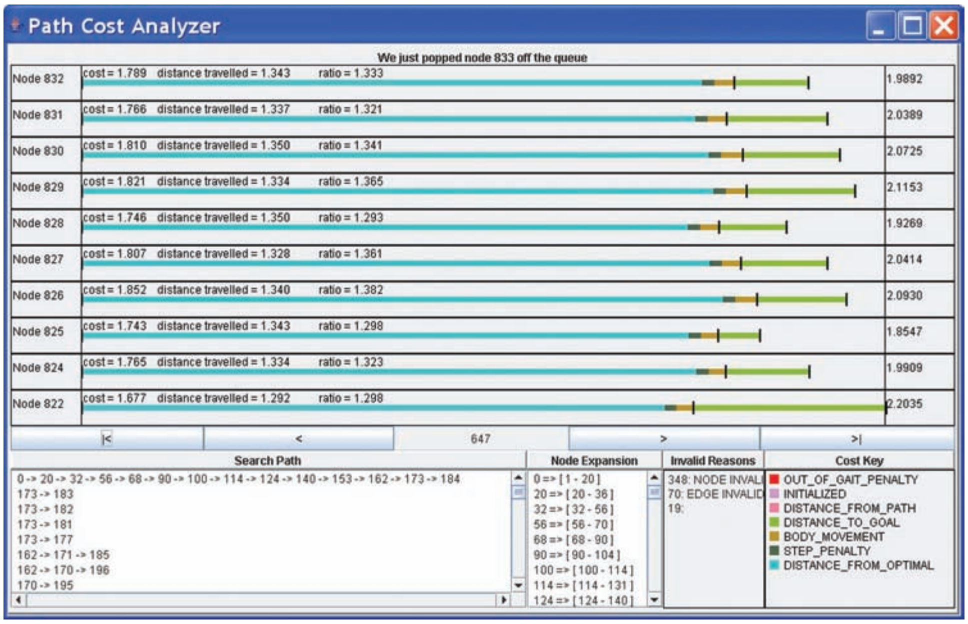
6.4. Fastest feasible trajectory

We implemented the Fastest Feasible Trajectory generator (Choset et al., 2005) in order to be able to playback body and swing trajectories while still obeying a given set of motion constraints. The motion constraints on the body included maximum velocities, accelerations, and maximum velocity close to support edges. The constraints on the swing leg were represented in the form of maximum velocity and acceleration of the motors (as opposed to the joints). The hip joints (pitch and roll) are connected to the two hip motors through a differential: the knee joint is directly controlled by the knee motor.

The swing trajectory and body motion represent a curve in nine-dimensional space. This curve can be parameterized by s, such that $s = 0$ is the start of the curve and $s = 1$ is the end of the curve. The Fastest Feasible Trajectory algorithm will solve for $s(t)$, such that s reaches 1 in minimum time while not violating any of the constraints. This algorithm is an iterative algorithm that tries to make sure that at least one of the constraints is at a limit such that the minimum time is guaranteed.

6.5. Visualization

Another key technique we used during the project was visualization. The complex and interrelated nature of the different components (terrain scoring, planning, execution) combined with the fast and dynamic nature of testing on



```
We just popped node 833 off the queue
Node 832  cost = 1.789  distance travelled = 1.343  ratio = 1.333  1.9892
Node 831  cost = 1.766  distance travelled = 1.337  ratio = 1.321  2.0389
Node 830  cost = 1.810  distance travelled = 1.350  ratio = 1.341  2.0725
Node 829  cost = 1.821  distance travelled = 1.334  ratio = 1.365  2.1153
Node 828  cost = 1.746  distance travelled = 1.350  ratio = 1.293  1.9269
Node 827  cost = 1.807  distance travelled = 1.328  ratio = 1.361  2.0414
Node 826  cost = 1.852  distance travelled = 1.340  ratio = 1.382  2.0930
Node 825  cost = 1.743  distance travelled = 1.343  ratio = 1.298  1.8547
Node 824  cost = 1.765  distance travelled = 1.334  ratio = 1.323  1.9909
Node 822  cost = 1.677  distance travelled = 1.292  ratio = 1.298  2.2035
```

```
Search Path
0 -> 20 -> 32 -> 56 -> 68 -> 90 -> 100 -> 114 -> 124 -> 140 -> 153 -> 162 -> 173 -> 184
173 -> 183
173 -> 182
173 -> 181
173 -> 177
162 -> 171 -> 185
162 -> 170 -> 196
170 -> 195
```

```
Node Expansion
0 => [ 1 - 20 ]
20 => [ 20 - 36 ]
32 => [ 32 - 56 ]
56 => [ 56 - 70 ]
68 => [ 68 - 90 ]
90 => [ 90 - 104 ]
100 => [ 100 - 114 ]
114 => [ 114 - 131 ]
124 => [ 124 - 140 ]
```

```
Invalid Reasons
348: NODE INVALI
70: EDGE INVALID
19:
```

Cost Key
- 🟥 OUT_OF_GAIT_PENALTY
- 🟪 INITIALIZED
- 🟥 DISTANCE_FROM_PATH
- 🟩 DISTANCE_TO_GOAL
- 🟨 BODY_MOVEMENT
- 🟩 STEP_PENALTY
- 🟦 DISTANCE_FROM_OPTIMAL

Fig. 21. The path cost analyzer was developed to be able to track how and why the planner made its decisions.

the real robot made debugging complex failures a challenge. Our solution was to implement visualizations on all critical aspects of the algorithm. We were able to visualize the results of ground scoring (Figure 5), coagulation (Figure 6), approximate body path for planning (Figure 7), footstep enumeration (Figure 11), path cost (Figure 21), pose finding, and swing trajectory. Our development environment also allowed us to rewind and playback simulations and actual runs and view results without opening a separate application, which was a tremendous benefit. We were also able to retain all this functionality in either simulation or the real dog without a change to the controller or code. Visualization at 10 ms resolution of an actual run using the actual robot was invaluable in debugging problems. To be able to rewind and slowly step through the events that ensued prior to a failure was a tremendously helpful tool.

Figures 23 and 24 show simulation results for a trial crossing the Round Rocks terrain. In Figure 23, the body center path is shown by the cyan dots. Note that it is generally smooth, except for one side shift due to a front left (yellow dot) step to the side. Figure 24 shows the swing leg timing for the same trial. The Xgait parameters, X_{FTD} and X_{HLO}, are automatically set by the controller based on the terrain and footsteps. Note that on smooth terrain, the parameters are set for a dynamic gait ($X_{\mathrm{FTD}} = 0.02$ and $X_{\mathrm{HLO}} = -0.005$). For rougher terrain, the parameters are set for a static crawl gait ($X_{\mathrm{FTD}} = -0.01$ and $X_{\mathrm{HLO}} = 0.01$). The yellow boxes show when a front leg is swinging, and the blue boxes show when a hind leg is swinging. Overlap between the yellow and blue (green boxes) are periods when two legs are swinging at the same time. Gaps between yellow and blue boxes are quad shifts, which are body shifts when all four legs are on the ground.

7. Results

Overall, our approach was highly successful. We consistently performed well during the program. The final test consisted of 10 different terrain configurations, with six of them unknown boards. These boards were similar to the ones shown in Extension 1. We crossed 28 out of 30 trials, achieving at least two successful runs out of three trials on every board. The metric speed for the program was 7.2 cm/s and we crossed eight out of 10 boards at greater than metric speed with an overall average speed of 11.2 cm/s. For the seven mandatory boards to cross, our average speed was 10.4 cm/s, which was the fastest average speed on these boards.

Figure 22 shows sequential images of the robot crossing the Logs terrain board. The speed for this trial was 8.3 cm/s.

8. Lessons learned

The L2 program resulted in one of the most significant development efforts on a given quadrupedal robot platform to date. From our effort on the L2 project, we have determined a few items in the hardware design of LittleDog that could significantly impact our development and performance. These items are listed below.

8.1. Force controllable actuators

Although there were load sensors in the feet, true force control algorithms, or compliant control, requires the ability to control the force or torque at every joint. Compliant control becomes more critical as the knowledge about the terrain decreases. In this project, the need for compliant control

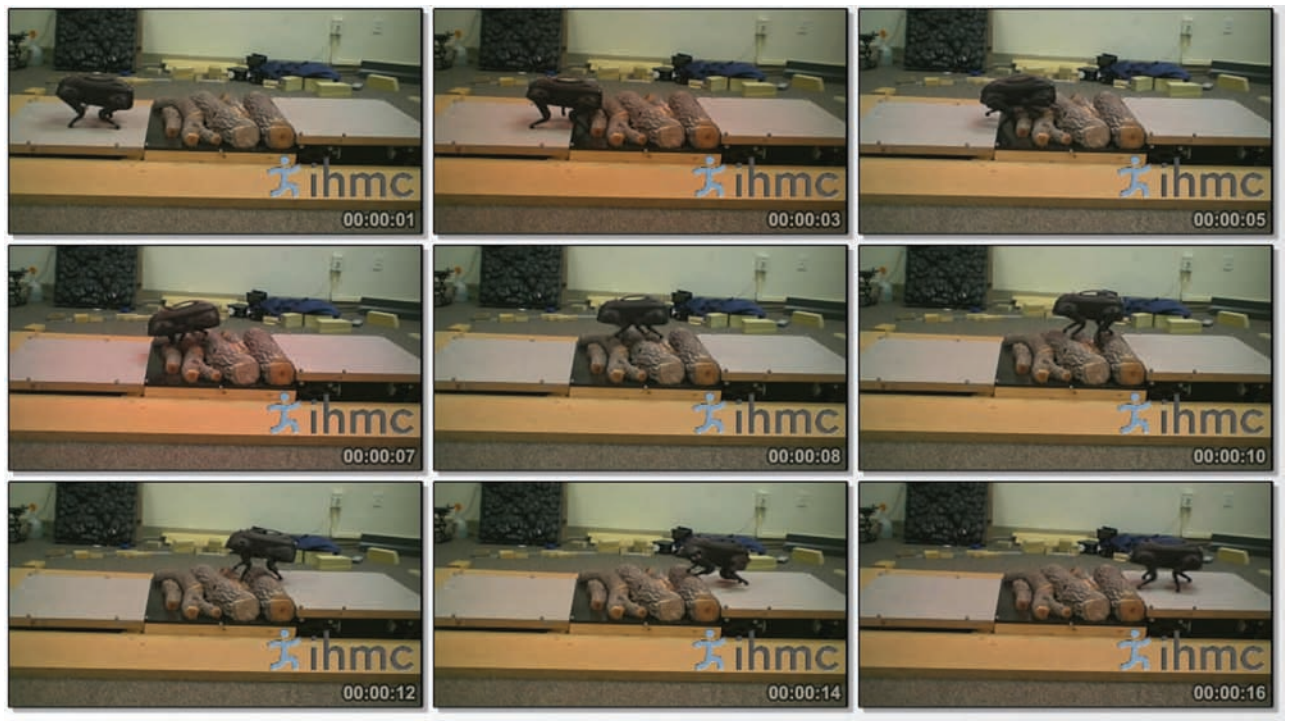


Fig. 22. Time sequences of LittleDog crossing the Logs terrain.

was not critical, because we had detailed knowledge of the terrain and real-time localization. However, force controllable actuators would have reduced or eliminated our main causes of falling. One of our biggest challenges was making sure not to swing a leg that was supporting weight. When a loaded leg is lifted, it causes the robot to fall until another leg becomes loaded. With force controllable actuators, ensuring adequate force distribution, and thereby preventing lifting a loaded leg, is a straightforward task.

8.2. Swing leg speed

The overall locomotion speed and robustness to slips and disturbances are directly related to the speed and accuracy in which a leg can be swung in relation to the height of the center of mass. For example, one of the best quadrupeds to react to disturbances is BigDog, from Boston Dynamics. This robot has a high center of mass and can swing its legs extremely fast. The center of mass of BigDog is about six times higher than that of LittleDog. Therefore, in order for LittleDog to have comparable reactive capabilities, its swing leg speed would need to be significantly faster than BigDog's. LittleDog's low center of mass and relatively slow swing speed limited the effectiveness of reactive behaviors and the dynamic maneuvers.

8.3. Hardware geometry

One issue on very rough terrain is being able to swing the leg forward without having to worry about leg collisions with the terrain. Because of the limited hip roll rotation,

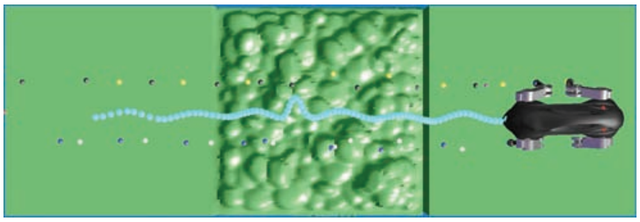


Fig. 23. Top view of body simulation results for crossing the Round Rocks terrain. The cyan dots show the body center path. The yellow, white, blue, and black dots show the planned footsteps for the FL, FR, HR, and HL feet, respectively. (Color online only.)

the elbow had to pass below the body on a forward swing motion. Another geometry issue was that the legs on the same side of the robot would collide knees for certain foot placement and body pose combinations. Although colliding combinations can be avoided in software, this limited some of the potential stances.

8.4. Machine learning versus engineered solutions versus planning

In the first phase of the project, we implemented some learning algorithms, such as off-line learning of terrain scoring, and learning robot parameters for better dynamical balance. In the second phase of the project, learning was less emphasized, and achieving the goal speed metrics was made the criterion for receiving the next round of funding.

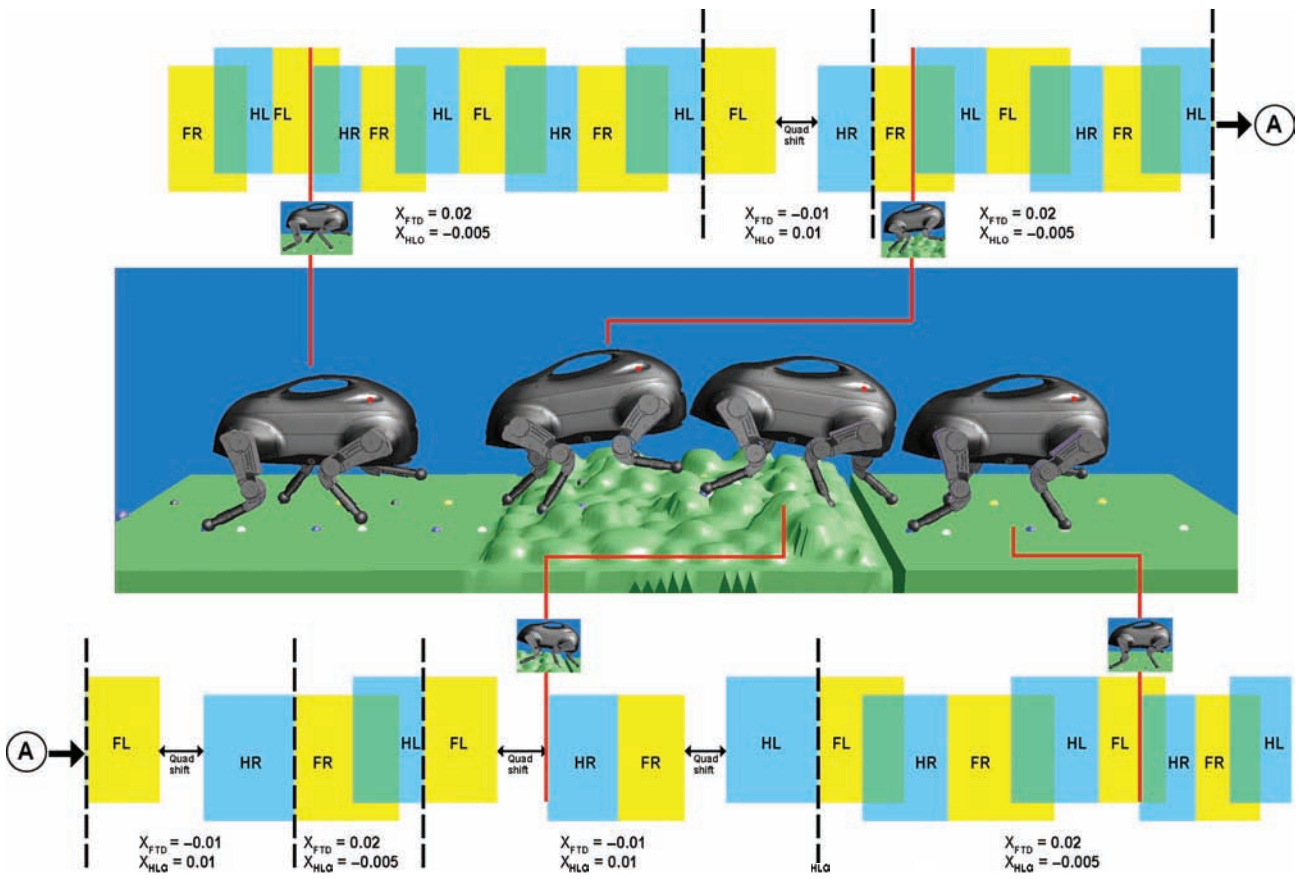


Fig. 24. This figure shows the swing leg sequencing for a simulated trial across the Round Rocks terrain. On the smoother parts of the terrain, the Xgait parameters are set such that the gait is dynamic with significant overlap between the front and following hind swing. For rougher parts, the Xgait parameters are set to generate a static crawl gait. (Color online only.)

Therefore, after phase I, we focused on engineered solutions, which were giving us very good performance, instead of learned solutions.

We believe that engineered solutions were competitive for this project, since it essentially was a high-dimensional geometric search problem without much uncertainty. Many learning techniques are essentially searches over large spaces with uncertainty in them. Planning techniques, on the other hand, are searches over large spaces without uncertainty in them. We, and all the L2 teams, did use extensive amounts of planning, and did use lots of search techniques in just about every module of our algorithms. However, since the uncertainty in the problem was small, these searches could all be performed off-line, or on full models, and did not improve with experience. Examples are as follows.

Learning the dynamics of the robot was not very critical. This was because the robot moved relatively slowly, and most of the mass was concentrated in the body. Therefore, a point-mass model was sufficient for any of the gaits we performed. In fact, a purely kinematic model of the robot was sufficient for most of the gaits.

Manually tuning the terrain cost function seemed to be sufficient for all of the boards. This was because we had sub-millimeter resolution of the terrain and simple heuristics could be determined that worked pretty well on the boards.

Learning anything about a terrain board that was not experienced before was difficult, since we were only given three runs per test on the ‘hidden’ boards. For the ‘known’ terrain boards we had plenty of time and near-perfect knowledge of the boards, such that learning novel situations was not required for them.

8.5. *Managing software complexity*

During the course of the project, we wrote over 1000 Java classes, some simple, some complex, some short (20 lines), and some long (of the order of 2000 lines). We quickly discovered that managing the complexity of the software was important and started to incorporate some software development best practices. We incorporated various tools and techniques, such as version control, unit testing, having the development team work in a single, open room, continuous integration, continuous refactoring, and daily short ‘stand-up’ meetings.

As each test approached, before any code could be committed to the repository, we required that it had to be checked by a team member who had not written it. Before any new set of software was used as the runtime software for a board on a government test, we would have it compete in a 'head to head' competition with the previous best algorithm for the given board. If it was not faster and/or more reliable we would not use it, but instead run the previous best algorithm for that board.

9. Applicability beyond LittleDog

In order to focus the study on quadrupedal locomotion, the L2 program management had to make some choices about things such as robot hardware, terrain type and scale, sensing, and localization in order to make the problem tractable, given the available resources. These restrictions naturally impact the direct applicability of the generated research concepts on areas beyond the L2 program. For example, the small size of LittleDog limits its use to appropriately scaled-down terrain. The availability of high-resolution terrain maps is something not typically available in real-world applications. Precise localization is also something that is often not available. Given the simplifications in the program, what results can and cannot be extended to applications beyond L2?

We feel that the terrain scoring work done on L2 will have limited direct applicability for a few main reasons. Firstly, many of the parameters are a function of the robot (e.g. foot size, joint configuration, range of motion). Second is that all of the L2 terrains were solid and had non-realistic friction. No terrain had compliance or plasticity. Thirdly, the algorithms that we developed for scoring required very precise knowledge of the local terrain height, including terrain heights that were occluded from the robot's direction of travel.

The planning work is certainly applicable beyond L2. While tailored to LittleDog, the general approach of using an optimal step is applicable to any quadruped. In addition, planning on moderately rough, non-maze-like terrain can be fast and done online.

The main contribution from our team that we feel is applicable beyond LittleDog is the Xgait. The fastest static crawl gait on rough ground that we were able to achieve with LittleDog was approximately 9 cm/s. With the Xgait, were able to walk at approximately 14 cm/s. The approach is general and can be applied to other quadrupeds to allow for a natural transition between crawl and trot gaits. Having a flexible gait that can adjust dynamically with the terrain will be a critical part of real-world quadrupedal locomotion.

A last area that we felt was not applicable was relying on highly accurate position localization relative to precise knowledge of the terrain. It is unlikely that real-world systems will have the accuracy afforded to the L2 project. Even with good accuracy and perception, some compliant control was desirable to manage ground contact and load distribution among the support legs. In real-world applications where this level of perception accuracy is unlikely, the importance of force control will be much greater.

Funding

This work was supported by the DARPA (contract number FA8650-05-C-7265).

Conflict of Interest

The authors declare they have no conflicts of interest.

Acknowledgements

This work was supported by DARPA as part of the L2 Program. This work benefitted from both competition and collaboration with the other L2 teams at MIT, Carnegie Mellon University, University of Pennsylvania, Stanford, and USC. Boston Dynamics developed and maintained the LittleDog robots, which proved to be a reliable platform for the project. Boston Dynamics provided one-day turnaround time for any robot maintenance or repair by having a fleet of well-tuned robots at the ready to be sent out to replace any down robot. This allowed the teams to focus on walking algorithms without being plagued by the downtime that is unfortunately typical of many research robots.

References

Buchli J, Kalakrishnan M, et al. (2009) Compliant quadruped locomotion over rough terrain. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Byl K, Shkolnik A, et al. (2008) Reliable dynamic motions for a stiff quadruped. In: *Proceedings of the 11th International Symposium on Experimental Robotics (ISER)*.

Choset H, Lynch KM, et al. (2005) *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press.

Craig JJ (2004) I*ntroduction to Robotics: Mechanics and Control*. New Jersey, USA: Prentice Hall.

Estremera J and Santos PGD (2002) Free Gaits for Quadruped Robots over Irregular Terrain. *The International Journal of Robotics Research* 115–130.

Estremera J and Santos PGD (2005) Generating continuous free crab gaits for quadruped robots on irregular terrain. *IEEE Transactions on Robotics* 21(6): 1067–1076.

Hirose S, Fukuda Y, et al. (2009) Quadruped walking robots at Tokyo Institute of Technology. *Robotics & Automation Magazine*, IEEE 16: 104–114.

Kalakrishnan M, Buchli J, et al. (2009) Learning locomotion over rough terrain using terrain templates. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hyatt Regency St. Louis Riverfront, St. Louis, USA.

Kolter ZJ, Abbeel P, et al. (2007) *Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion*. Cambridge, MA: MIT Press.

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

Oomichi T, Fuke Y, et al. (1994) Navigation of a quadruped robot in uneven terrain with multiplefoot sensors. In: *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*.

Pack DJ and Kang H (1999) Free gait control for a quadruped walking robot. *Laboratory Robotics and Automation* 11(2): 71–81.

Prajoux R and Martins LdSF (1996) A walk supervisor architecture for autonomous four-legged robots embedding real-time decisionmaking. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Song S and Waldron KJ (1998) *Machines That Walk: The Adaptive Suspension Vehicle*. Cambridge, MA: MIT Press.

Appendix: Index to Multimedia Extensions

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

Extension	Media Type	Description
1	Video	This video provides an overview of the L2 project and describes IHMC's approach to the project. The video concludes with a compilation of the 10 terrains used of the final evaluation by the DARPA.