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

Autonomous Navigation for BigDog

David Wooden*, Matthew Malchano*, Kevin Blankespoor*,
Andrew Howard†, Alfred A. Rizzi*, and Marc Raibert*

* Boston Dynamics
78 Fourth Avenue, Waltham, MA 02451
{dwooden, malchano, blankespoor,
arizzi, mxr}@bostondynamics.com

† Jet Propulsion Laboratory
Pasadena, CA 91109
andrew.howard@jpl.nasa.gov

***Abstract*— BigDog is a four legged robot with exceptional rough-terrain mobility. In this paper, we equip BigDog with a laser scanner, stereo vision system, and perception and navigation algorithms. Using these sensors and algorithms, BigDog performs autonomous navigation to goal positions in unstructured forest environments. The robot perceives obstacles, such as trees, boulders, and ground features, and steers to avoid them on its way to the goal. We describe the hardware and software implementation of the navigation system and summarize performance. During field tests in unstructured wooded terrain, BigDog reached its goal position 23 of 26 runs and traveled over 130 meters at a time without operator involvement.**

I. Introduction

BigDog is a rough-terrain quadruped robot that walks, runs, climbs and carries heavy loads. BigDog is powered by an internal combustion engine that drives a hydraulic actuation system. Its legs include compliant elements near the feet to help absorb shock and regulate ground forces. BigDog is the size of a large dog or small mule, about 3 feet long and 2.5 feet tall, and weighs 240 lbs.

We describe the application of autonomous navigation techniques to BigDog. We demonstrate the ability to navigate unstructured outdoor environments by taking advantage of BigDog's rough-terrain mobility and by introducing exteroceptive sensors capable of sensing terrain. Prior to this point, research on the BigDog platform had primarily focused on developing mobility control for challenging terrain and hardware rugged enough for real-world field operation.

Fig. 1: The BigDog robot.

Prior to this work, an operator needed to continually steer the robot by specifying steering and forward speed commands while BigDog stabilized itself automatically over rough terrain. The operator was responsible for ensuring the robot did not collide with obstacles while directing it towards a goal. In this paper, we describe how we can replace the operator with an autonomous system that uses sensing, mapping, and planning techniques and exploits BigDog's robust gait control.

The navigation system uses a combination of planar laser scans, stereo vision, and proprioceptive sensing to accomplish a number of tasks. It estimates the position of BigDog in the world. It perceives certain types of obstacles and places them into a 2D representation of the world. It then plans paths and steers the robot to follow them. The path planner is a variation on the classic A^* path planner [5]. A spline smoothing algorithm [1] smooths these paths and passes them into a path follower. The path follower then calculates steering commands to drive BigDog along the planned path.

Other mobile robots employ similar approaches utilizing the common technique of planning paths over grid-based models of the terrain [3]. These robots include DARPA Urban Challenge vehicles [14], [11], the DARPA Learning Applied to Ground Robots (LAGR) program, and the DARPA UPI Crusher program, among many others. However, since the vast majority of ground robots are either wheeled or tracked vehicles, these algorithms and approaches have not been tested on legged platforms yet.

The system described was tested in a forest area populated with trees, boulders, saplings, up to 11° slopes, and other features typical of New England woodlands. A total of 26 separate test runs was executed, of which 88% reached the goal. The robot navigated up to 130 meters in a given run without failure and traveled over 1.1 km while conducting these tests.

The organization of this paper is as follows. In Section II, we describe the hardware platform of BigDog and formulate the sensing and navigation problem. In Section III, we describe the methods employed for modeling the environment, computing smooth paths to the goal over that model, and controlling the robot to track those paths. In Section IV, we describe some of the field testing done to demonstrate the capability of the system. In Section V, we discuss future work and draw conclusions.

II. Platform Details and Problem Formulation

Sensors are used to detect obstacles and terrain near the robot, determine the robot's body position and orientation,

and estimate the state of the robot's joint angles or forces. This section describes these sensors and the computing hardware used to control the robot. In addition, we formulate the autonomous navigation problem we set out to solve.

A. Hardware

An introduction to BigDog including its hardware systems is given in [12], [13]. The following supplements that introduction.

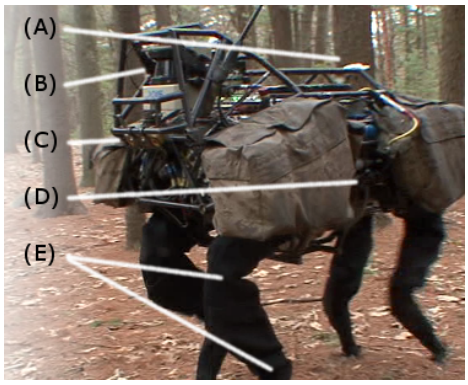


Fig. 2: Depiction of BigDog with sensors called out: (a) GPS antenna (b) SICK LIDAR (c) Bumblebee stereo camera pair (d) Honeywell IMU (e) Joint sensors.

1) Proprioceptive Sensors: BigDog hosts a variety of proprioceptive sensors used for gait control and autonomous navigation. Each of the robot's sixteen active and four passive degrees of freedom is instrumented to provide position and force measurements. These measurements are combined with data from an IMU to obtain estimates of various quantities such as ground contact state, ground plane orientation, body velocity, and world-frame pose needed for both gait and navigation control. Additionally, a variety of sensors provide status for the power, computation, hydraulic, thermal, and other systems necessary for BigDog operation.

2) Exteroceptive Sensors: The robot is equipped with four exteroceptive sensors (see Figure 2): A SICK LMS 291 LIDAR scanner, a PointGrey Bumblebee stereo camera, a NovAtel GPS receiver, and a Honeywell IMU. The raw sensory data is the input to the system architecture depicted in Figure 4.

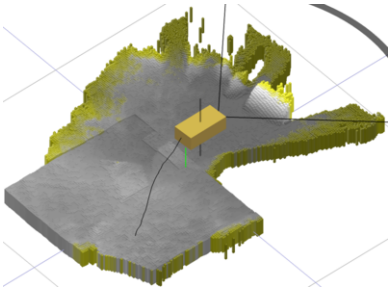


Fig. 3: Side view of 3D terrain map generated by the stereo vision system [6], as the robot approaches a group of boulders. Cells are colored yellow if they have been observed a small number of times and grey if they have been observed a large number of times.

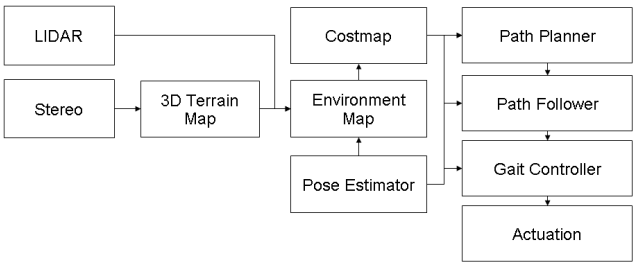


Fig. 4: System software architecture. LIDAR and 3D Terrain Data (from stereo images) combine to form a spatiotemporal map of the environment. From this map we generate a costmap, over which we plan a path. The path follower feeds body velocity commands to the gait control component.

3) Computers: Two computers are used to implement the system shown in Figure 4. BigDog's *main computer*, is a PC104 stack, with a single-core Intel Pentium M CPU (1.8 GHz). It interfaces with the proprioceptive sensors, controls the robot's balance and motion, computes an up-to-date model of the robot's environment, calculates a path through that environment, and executes gait control. A separate *vision computer*, running on an Intel CoreDuo CPU (1.7 GHz), communicates with the camera pair, computes stereo disparity maps and visual odometry estimates, and maintains a 3D terrain map of the ground near the robot [6]. The vision computer communicates the map and visual odometry position estimates to the main computer at 15Hz via onboard ethernet.

B. Problem Formulation and Assumptions

The BigDog gait control system provides reliable rough terrain mobility over a wide variety of challenging terrains: sloped terrain, scree, rubble, mud, vegetation, etc. The autonomy and perception system described in this paper provides steering commands to the gait control system based on paths it plans around objects such as logs, trees, and boulders. A strength of the gait control system's reliability is that it enables us to simplify the planning problem to be two dimensional and discrete, consisting of traversable and untraversable regions. In other words, even though the data provided by the LIDAR and stereo sensors is three dimensional, we are able to rely on the self-stabilization of the gait control system to forgo a more complex 3D perception and planning problem.

III. Technical Approach

Our general technical approach makes use of data from two environmental sensors to identify obstacles, compute a trajectory through or around the obstacles, and command the gait control system to track that trajectory.

This overall process can be broken down into three steps. First, raw LIDAR scans and camera images are processed to produce lists of points in the world frame that indicate obstacles in the environment. These points are then segmented into disjoint objects and tracked over time. Second, these objects are combined in a temporal memory which is used to construct a cost map of the environment surrounding the robot. This cost map is then used to plan a path to an intermediate goal. The planner is designed to ensure that

paths keep BigDog an appropriate distance from obstacles and that the paths are spatially stable over iterations of the planner. The robot's path-following algorithm causes the robot to follow paths by sending body velocity commands to the gait control system, which in turn moves the legs to travel the specified path.

A. Derived Sensing

1) Pose Estimation: There are two sources of odometry information, the kinematic sensors in the legs and the vision system. Odometry from these two sources are fused to generate a single robot pose estimate (see Figure 5). An estimate of BigDog's 6 degree of freedom pose is used to integrate sensory data into the map and to estimate the robot's location on the map.

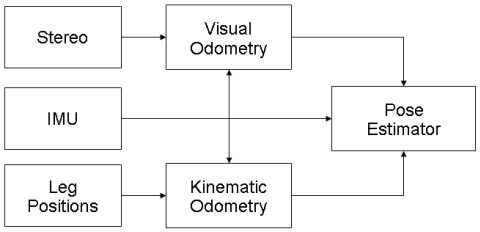


Fig. 5: Pose estimation architecture. Kinematic and visual odometry along with an IMU unit are fused to produce a 6DOF pose estimate.

The kinematics-based odometry system uses kinematic information from the legs that are in contact with the ground to estimate robot motion, while the visual odometry system tracks visual features to estimate robot motion. Both use the IMU as a source of orientation information. The overall pose estimator combines the outputs of these two odometers, generally emphasizing visual odometry at low speeds and kinematics at higher speeds. The fusion of the two is intended to address specific weaknesses of each estimator: drop-out of stereo, drift of the legged odometer while trotting in place, and vertical-axis errors of legged odometry. We estimate that the pose estimator's world position drifts with respect to ground truth at a rate of 0.005 to 0.010 meters of drift per meter of travel.

The LIDAR sensor used on BigDog provides a new scan every 13ms. Each scan is transformed into a world coordinate frame centered on the robot's position using time-synchronized information from the pose estimator. The resulting 3D point cloud is then handed to the segmentation algorithm, described below. Similarly, the stereo vision system accumulates disparity maps over time to maintain a 3D terrain map of the environment in a 4m by 4m square centered on the robot. A spatial filter identifies regions of substantial height variation (*i.e.* potential obstacles) and passes a list of points belonging to those regions to the point cloud segmentation algorithm.

2) Point-cloud Segmentation and Object Tracking: Due to the slope of the ground and the motion of the robot's body, portions of the LIDAR scanner's data will include ground scans. Similar in appearance to ground returns are the returns that come from long obstacles (*e.g.* walls). To be successful, the system must interpret these returns so as to navigate around walls, but not appear to be afraid of the ground. Our first step in this process is the segmentation of the obstacle points provided by the LIDAR scanner and stereo-based terrain map into distinct objects. Sparse 3D point clouds are segmented into objects by merging individual points that are separated by less than 0.5 meters.

Objects produced by the segmentation algorithm are tracked over time. We employ a greedy iterative algorithm with heuristic constraints to accomplish this task. An object in the current scan is matched with the nearest unmatched object of the last scan, provided that the objects are separated by less than 0.7 meters. With the point clouds segmented into objects and tracked over time, the robot is able to behave appropriately in an environment with moderate ground slope variation and obstacles of various types: trees, boulders, fallen logs, walls. Trees and walls are identified primarily by the LIDAR scanner, and boulders and fallen logs are identified by the stereo vision system. See Figure 6.

B. Navigation Planning

We present an approach to the navigation problem for BigDog that is common in the robotics community. Obstacle points (generated by the derived perception processes) are deposited into a cost map centered on the robot's position. The robot's ultimate goal point is projected onto the boundary of the cost map, and a variant of the A^* algorithm is executed over this map. This process repeats approximately once a second.

1) Memory of tracked obstacles: Because of the limited field of view of the robot's two sensors, it is critical that the robot keep an accurate memory of obstacles that it can no longer see. As object lists are provided by the *object tracker*, individual objects are added, updated, or removed in the planning system's object memory. The size of the list of objects kept by the planning system is bounded so as new objects are appended, other objects must be removed.

Given the current list of objects $\mathcal{O}$, we can compute two parametrized subsets of $\mathcal{O}$:

$$P(t) = \Big\{ q \in \mathcal{O} \;\Big|\; \mathrm{age}(q) > t \Big\},$$

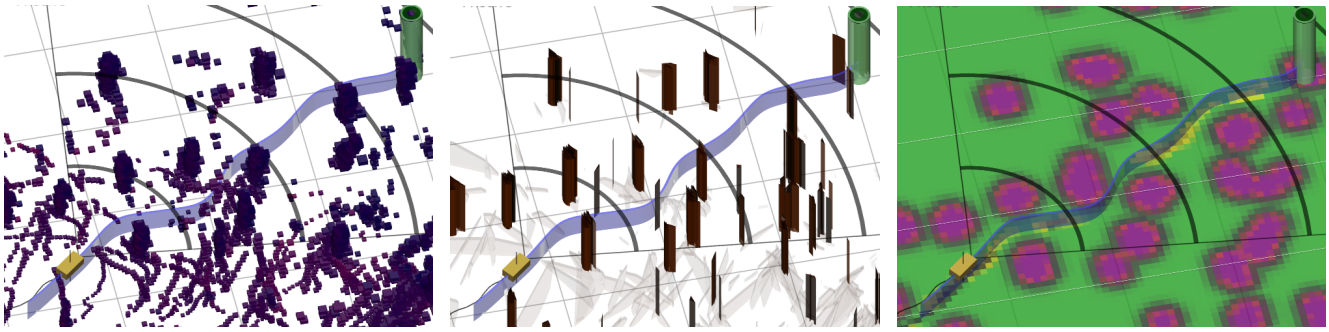
and

$$Q(d) = \Big\{ q \in \mathcal{O} \;\Big|\; \mathrm{norm}(q, r)_{\mathrm{inf}} > d \Big\},$$

where $\mathrm{age}(q)$ is the time difference between the current time and the time that the object q was last measured, and $\mathrm{norm}(q, r)_{\mathrm{inf}}$ is the minimum distance between the current position of the robot and boundary of the object q.

Objects are removed from $\mathcal{O}$ according to the following criteria:

- The set $\{P(30) \cap Q(15)\}$ is subtracted from $\mathcal{O}$. That is, objects 30 seconds old or older and at least 15 meters from the robot are forgotten.
- The set $\{P(1800) \cap Q(10)\}$ is subtracted from $\mathcal{O}$. That is, objects half an hour old or older and at least 10 meters from the robot are forgotten.
- Finally, objects are dropped from $\mathcal{O}$ until the list size is below its limit, prioritizing objects by the amount of time they were successfully tracked by the *object tracker*. In other words, the objects that have been seen longer are preferentially kept in memory.



(a) (b) (c)

Fig. 6: A progression of figures showing the robot (yellow box) with (a) several seconds of raw LIDAR data (blue dots) and (b) the corresponding *objects*. Tree objects are shown tall and brown; ground returns are shown transparent and flat. The path (blue ribbon) to the goal (green cylinder) are also shown. (c) Overhead view of the costmap with the following coloring: green areas are nominal cost, purple/red areas are high cost, and yellow areas are low cost. The grid subdivision is 5 meters.

- However, no objects detected in the previous 10 seconds are discarded.

This allocation of memory resources leads to the following behavior: as objects pass out of view of the robot's sensors, the robot forgets objects that are far away and forgets objects that it has not seen many times (and which might actually be spurious). Objects that are in view or that are out-of-view but close to the robot are not forgotten.

2) Costmap Generation: We use a costmap constructed over a 2D grid to represent the environment near the robot. The value (or cost) of a cell in the costmap is directly related to the intraversability of that cell for the robot.

Rather than dynamically maintain a costmap (as the robot perceives progressively more of its environment), a new costmap is generated each planning iteration and populated with the objects kept by the planner's memory. This implies that a dynamic path planner (e.g. see [4], [8], [10]) cannot be used in place of A^*. Because we assume the size of objects is bounded (that no cul-de-sac in the environment is larger than half the width of the costmap), the scope of the planning problem and the time needed to compute a path over the costmap is small.

The costmap is populated with cost from the list of objects according to the following algorithm:

Algorithm 1: Population of costmap from objects.

```
1 foreach q in O do
2     foreach point p in object q do
3         (i, j) = cell location belonging to p
4         set cost at (i, j) to be at least lethal_cost
5         foreach cell c near (i, j) do
6             set cost at c at least f(norm(c,(i,j)))
7         end
8     end
9 end
```

The cost of cells that belong to object points are assigned a very high *lethal cost*. The cost of cells near object points are set according to a function f of the distance from that cell to the object point. For the test results presented in this paper, f was simply the inverse cube of the distance. See Figure 6c.

The effect of this approach is that cells in the costmap where objects are given very high cost and cost falls off smoothly in the neighborhood of these cells.

3) Path stability: To ensure that we do not "drive" BigDog in a haphazard manner, special care is taken to ensure that the planned path is as stable as feasible over iterations of the path planner. This is done in three ways.

First, the starting point given to the A^* algorithm is not the robot's current position r, but rather the projection of the robot's position onto the last path output by the A^* algorithm (call this point p). As BigDog follows a planned path, it tends to wander laterally with respect to the path. By projecting the start point onto the previous A^* path, the vacillation of the robot's body position is effectively filtered out and the output paths of the path planner tend to be more stable. In the event that the robot deviates more than a user-specified distance (currently set to 3m) from the last planned path, p is simply set to robot's current position.

Second, to ensure continuity of the path follower (see III-C.1), we compute q, the projection of the robot's position from 2.5 seconds in the past onto the last plan output by A^*. Then, the section of the last planned path from q to p is prepended to the output of newest planned path. This tends to maintain a small amount of path behind the robot as it drives towards the goal. Path continuity aids in making the output of the path follower better behaved in the presence of substantial position disturbances that are not uncommon on a legged robot.

Third, a small history of planned paths is kept. These paths are used to discount cells of the costmap where the robot has previously planned to go, with very heavy discounting applied to the area near the robot. This tends to make the next planned path follow the previous planned path (but without strictly guaranteeing it).

4) Path smoothing: The path output by planning over a regular grid is necessarily jagged. The large changes in heading in this path can induce undesirable steering commands. To correct this, we apply a deBoor smoother to the output of the path planner (see [1]).

In addition, planning paths over a grid of uniform cost

often leads to technically optimal but less than desirable paths to the goal [4]. We address this issue by computing a grossly over-smoothed path each iteration of the path planner. On the subsequent iteration, costmap cells near the over-smoothed path are assigned a reduced cost. Over successive iterations of the path planner, this process tends to provide a straighter and smoother collision-free path to the goal.

C. Gait Control: Mobility, and Balance

The navigation planning system produces a new path approximately once per second. This path consists of a smoothed spline sampled at a high density relative to the geometric information in the path. The path following algorithm, which runs at 200Hz, steers the robot in an attempt to track the most recent planned path. The path follower produces a set of steering commands in the form of desired body velocities, including forward speed, lateral speed, and yaw-rate of the body. These desired velocities are passed to the gait controller which controls the motion of the legs to achieve these velocities.

1) Path follower: paths to body velocities: Based on the distance between the robot and the path, one of three types of control strategies is used. If the robot is near the path segment, it is steered to move parallel to the tangent of the path and commanded to laterally side-step onto the path while traveling forward at full speed. If the robot is far from the path, it is steered directly toward a target point on the path. In a region between near and far, a convex combination of these strategies is used.

2) Gait control: body velocities to actuation: A detailed description of the gait control algorithms are beyond the scope of this paper. However, in general, body velocities act as control inputs to BigDog's low-level gait controllers. The gait controller produces force and position commands for every joint in order to assure stability, react to disturbances, and satisfy the desired body velocities. Although the path follower outputs can be used by any gait on the BigDog system, the Trot gait (in which legs are paired diagonally in the stance and swing phases) is the most suitable gait for the application described in this paper due to its blend of speed and rough-terrain capability

IV. Field Test Results

The sensing and navigation system described in this paper was installed on BigDog and tested in an unstructured outdoor environment. Tests were conducted at a large nearby outdoor park populated with trees, boulders, saplings, hills (up to 11° slope) and other features typical of a temperate forest. Figures 7 and 8 illustrate some of the variation encountered at this location. Figure 9 depicts the raw LIDAR data processed by the robot; dark areas represent trees and lighter areas represent ground returns – in the area shown, the robot is climbing a slight hill.

A. Test Results

The navigation sensing and planning software was developed over a period of seven months, with periodic field testing occurring approximately once every five weeks. Here, we describe the most recent experimental evaluation we conducted over the course of one day.

The robot traveled about 130 meters in individual autonomous runs without external influence by an operator. A total of 26 separate test runs were executed, 23 of which the robot reached the goal and had no collisions or near collisions with an obstacle. These runs are tagged in Table I as *Goal*. The robot fell at the end of only one run after stepping onto a low rock that the Gait Control system is typically able to traverse, but which in this case it did not (tagged as *Fall*). In three tests the robot encountered a "large" obstacle (greater in width than 20 meters) causing the robot to alternate between planning a path around either side of the obstruction without making sufficient forward progress in a predetermined amount of time (20 seconds). Obstructions of this size are specifically outside of the scope for which the autonomy system was designed, but which the robot encountered nonetheless. These instances are tagged in Table I as *Live-lock*.

The robot was placed in a small variety of scenarios,

Fig. 7: BigDog dodges boulders before climbing hill.

Fig. 8: BigDog during autonomy distance testing.

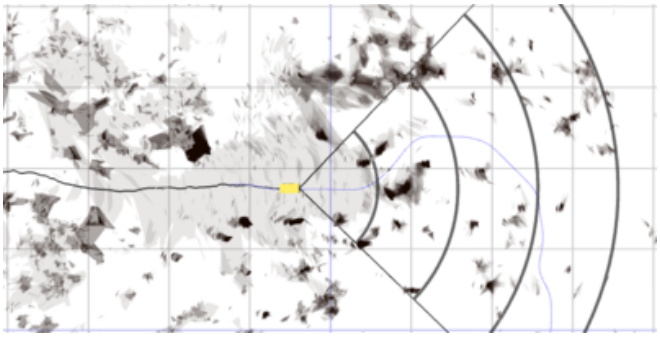


Fig. 9: Overhead view of segmented LIDAR and stereo-based objects in the middle of a test run. Dark regions indicate trees and other obstacles. Light regions indicate areas treated as ground returns. Grid subdivision is 5 meters.

ranging from open flat treed areas, to areas with boulders and trees, to areas with moderate ground variation (including ground slopes above 11°) with saplings and other undergrowth. As the robot was evaluated against increasingly difficult terrain and areas where our original assumptions had to be relaxed, the robot exhibited more *Live-lock* behavior and took less efficient paths. Table I shows the total path length, end-to-end distance, total run time, and run outcome of each test. The accumulated path length was about 1,100 meters. The median path length and run time per run was 36 meters and 61 seconds, respectively. Overall, this system was shown to be quite successful at navigating the unstructured wooded area we put the robot in, with the exception of the conditions for which the system was explicitly not designed to address.

ID	Path Length (m)	Point Dist. (m)	Run time (s)	Result
1	–	–	–	Goal
2	39.95	28.01	66.0	Goal
3	34.30	29.68	53.4	Goal
4	44.06	32.27	77.3	Goal
5	36.00	27.45	60.9	Goal
6	34.51	32.11	46.1	Goal
7	34.07	31.56	45.6	Goal
8	42.14	38.52	64.1	Goal
9	43.77	35.96	64.8	Goal
10	57.24	32.00	113.5	Goal
11	73.45	68.91	101.1	Goal
12	138.04	131.51	178.0	Live-lock
13	40.47	39.08	66.6	Goal
14	71.19	40.97	163.4	Goal
15	12.57	9.32	28.8	Goal
16	19.67	17.89	29.1	Goal
17	37.06	31.64	55.3	Goal
18	7.65	7.07	15.2	Goal
19	12.39	10.10	26.1	Live-lock
20	21.84	9.58	53.9	Goal
21	39.07	34.09	89.5	Goal
22	2.05	1.43	9.2	Goal
23	24.11	20.65	55.7	Goal
24	19.43	9.85	41.5	Goal
25	62.25	28.86	144.5	Goal
26	145.27	78.84	324.2	Live-lock
27	28.96	15.40	62.0	Fall

TABLE I: Tabulation of path length, end-to-end distance and run time for each test of the autonomy system. Records filled with "–" were regrettably lost or corrupted.

V. Conclusions and Future Work

This paper describes the application of autonomous navigation techniques to a rugged, outdoor quadruped robot. Included is a brief introduction to the component modules, as well as data from experimental field evaluation of the system.

More thorough test and evaluation of BigDog's navigation system is still to be done, as well as a complete analysis of how and when this system succeeds and fails in general outdoor environments. In addition, none of the sensing and planning approaches described in this paper exploit the unique properties of a robot that has legs instead of tracks or wheels. Problems such as precise foothold selection, footfall planning, and incorporation of terrain sensing in leg control are clear candidates for future work to boost the autonomous capability of quadrupedal and bipedal robots. Of special note is the work done for the DARPA Learning Locomotion program [2], [7], [9].

The mapping and planning techniques described by this paper were purposefully designed to describe the environment as two dimensional. This was made possible by the robust mobility capable of BigDog's gait control as well as the assumed benignity of the world the robot would encounter: a generally flat world populated with trees and boulders. A 2D model of the world simplifies the effort needed to maintain an accurate memory of the environment outside the robot's immediate field of view, particularly in the face of position estimation error. An alternative would be to model the world as three dimensional explicitly, allowing for direct and precise measurement of more challenging terrain (*e.g.* ground slopes of up to 35°). This information about the slope and quality of the terrain would be incorporated into the navigation system, affecting the planned path and the commanded velocity. However, this would also require additional computational resources not currently available on the BigDog platform.

VI. Acknowledgements

We thank the BigDog team, whose hard work and engineering were critical, along with the computer vision group at the Jet Propulsion Laboratory. The autonomy work reported was sponsored by TARDEC under contract W56HZV-05-C-0254. The BigDog project is funded by the Defense Advanced Research Projects Agency through contract N66001-07-C-2029, with additional funding from the US Marine Corps.

References

[1] C. D. Boor. *A practical guide to splines*. Springer-Verlag, 1978.
[2] K. Byl. *Metastable Legged-Robot Locomotion*. PhD thesis, Massachusetts Institute of Technology, September 2008.
[3] D. Ferguson, M. Likhachev, and A. Stentz. A guide to heuristic-based path planning. In *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling*, June 2005.
[4] D. Ferguson and A. Stentz. Multi-resolution field D*. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, March 2006.
[5] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, pages 100–107, 1968.
[6] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Proc. of the IEEE Int. Conf. on Intell. Robots and Systems*, pages 3946–3952, 2008.
[7] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal. Learning locomotion over rough terrain using terrain templates. In *IEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.
[8] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *Transactions on Robotics and Automation*, 2005.
[9] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng. A control architecture for quadruped locomotion over rough terrain. In *IEEE International Conference on Robotics and Automation*, 2008.
[10] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, , and S. Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *Int. Conf. on Automated Planning and Scheduling*, 2005.
[11] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The stanford entry in the urban challenge. *J. Field Robot.*, 25(9):569–597, 2008.
[12] R. Playter, M. Buehler, and M. Raibert. Bigdog. In *Proc. SPIE*, volume 6230, May 2006.
[13] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, and T. B. Team. Bigdog, the rough-terrain quaduped robot. In *International Conference of Automatic Control World Congress*, 2008.
[14] C. Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(1):425–466, June 2008.