In Proceedings, IEEE Conference on Robotics and Automation, April 2000, San Francisco.

Recent Progress in Local and Global Traversability for Planetary Rovers

Sanjiv Singh, Reid Simmons, Trey Smith, Anthony Stentz, Vandi Verma, Alex Yahja & Kurt Schwehr

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Autonomous planetary rovers operating in vast unknown environments must operate efficiently because of size, power and computing limitations. Recently, we have developed a rover capable of efficient obstacle avoidance and path planning. The rover uses binocular stereo vision to sense potentially cluttered outdoor environments. Navigation is performed by a combination of several modules that each "vote" for the next best action for the robot to execute. The key distinction of our system is that it produces globally intelligent behavior with a small computational resource— all processing and decision making is done on a single processor. These algorithms have been tested on our prototype rover, Bullwinkle, outdoors and have recently driven the rover 100 m at speeds of 15 cm/sec. In this paper we report on the extensions on the systems that we have previously developed that were necessary to achieve autonomous navigation in this domain.

1 Introduction

Unstructured outdoor environments, such as the surface of planets, pose special challenges for autonomous robots. Not only must the robot navigating in such environments avoid colliding with obstacles such as rocks, it must also avoid falling into a pit or ravine and avoid travel on terrain that would cause it to tip over.Vast areas often have open spaces where a robot might travel freely and are sparsely populated with obstacles. However, the range of obstacles that can interfere with the robot's passage is large— the robot must still avoid rocks as well as go around hills. Large areas are unlikely to be mapped at high resolution a priori and hence the robot must explore as it goes, incorporating newly discovered information into its database. Hence, the solution must be incremental by necessity.

Another challenge is dealing with a large amount of information and complex vehicle dynamics. Taken as a single problem, so much information must be processed to determine the next action that it is not possible for the robot to perform at any reasonable rate. We deal with this issue by using a layered approach to navigation. That is, we decompose navigation into two levels— local and global. The job of local planning is to react to sensory data as quickly as possible avoiding hazards of various kinds. A more deliberative, global process, operating at a coarser resolution of information determines how to steer the vehicle such that it can get to the goal, sometimes deciding to temporarily move away from the goal to reach the final destination. Both local and global schemes evaluate traversability. The local planner evaluates how certain it is of the shape and the "goodness" (suitability for travel) of the terrain just ahead of the vehicle, while the global planner evaluates the cost of traversal to the goal based on a combination of metrics.

This approach has been used successfully in the past in several systems at Carnegie Mellon. This paper reports on recent work that uses the a similar philosophy to autonomously navigate a planetary rover. Operation is such environments poses two additional challenges. First, since the surface might be very cluttered, it is important for the system to discriminate among obstacles that should be avoided if there is a choice versus those obstacles that must be avoided at all costs. A conservative planner that regards all detectable objects as obstacles will not exploit the ability of the rover to drive over some obstacles. Second, planetary rovers carry very limited computing and thus it is important that sensor processing and decision making are as efficient as possible. We have developed a system based on a small outdoor robot that navigates in cluttered as well as open environments at ground speeds surpassing the targets set for upcoming rover missions. The rover uses a modest (relative to the state of the art) single processor for all sensing and decision making. Here we discuss the overall system architecture and report on results from simulation as well as from experiments with our rover, Bullwinkle. Currently, we are in the process of porting our software system to run on the next generation Mars Rover, being developed at JPL (Fig. 1).

2 Related Work

Nearly all the research in local obstacle avoidance for indoor robots, and much of the work in outdoor vehicles, has used the assumption that the world is composed of obstacles and free space [1],[4],[10]. As described above, this assumption has serious consequences for navigation in rugged terrain, where sometimes the best (or only) choice is to surmount low obstacles, or to travel a short distance through somewhat rough terrain, rather than tak-

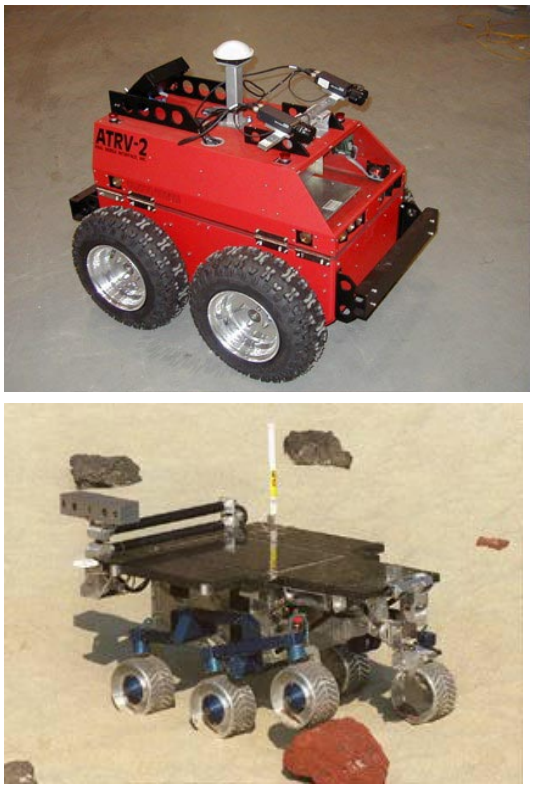

Fig. 1 Rovers. (top) CMU prototype rover based on RWI's ATRV2, Bullwinkle. (bottom) Next generation planetary rover developed by NASA JPL. Bullwinkle is equipped with stereo cameras, gyro and tilt sensors. All computing is performed on a single, low-power processor intended to emulate the processing that will be available on the planetary rover.

ing a long detour.

Recently, however, some researchers have investigated use of continuous measures of traversability in order to enable vehicles to make such decisions. The Ranger algorithm [5], on which our original navigation system was based [14], performed a simulation of the vehicle through the terrain, and analyzed the roll, pitch and high-centering of the vehicle as it moved along predetermined arcs. We have moved to a more statistical analysis of traversability, in part to deal better with sensor noise and in part to facilitate the merging of data over time— by merging traversability maps rather than terrain maps we reduce the effects of dead-reckoning error.

The work of Seraji uses linguistic terms (e.g., "passable", "highly-impassable") to represent traversability measures [13]. Fuzzy logic, in the form of expert-system-like rules, are used to decide how to move and turn to avoid obstacles. Several methods for producing the traversability measures from actual data are suggested, but it is not clear how the system (which was tested in simulation) actually calculated these terms. Gennery has proposed a method of traversability analysis for a planetary rover very similar to ours [6]. His method, like ours, fits planes to small terrain patches and uses the plane parameters and plane-fit residual to estimate slope and roughness, respectively. The major difference is that Gennery's algorithm is iterative, and thus more computationally complex than ours, and our algorithm computes residual at two levels, and can thus differentiate roughness at different scales. Also, Gennery's system does path planning at one level of granularity, while our navigation system divides the work into a local and global planners.

Global navigation is the task of moving the rover from some start location to a goal location. For the application considered, the environment is presumed to be unknown or partially-known. Given the limitation on prior information, the rover cannot pre-plan a path that is optimal and guaranteed to reach the goal. Instead, the rover acquires sensor information about the environment as it navigates and modifies its plan accordingly. One approach is to combine directed navigation with undirected exploration to learn a good route to the goal [7],[12],[20]. This approach is most appropriate for environments that are traversed multiple times, since the rover discovers better routes over time through trial and error. A second approach is to attempt to drive directly to the goal, circumnavigating obstacles along the way [8],[9].This approach requires little state information and is easy to implement. However, the approach does not make use of prior information, and it assumes a binary world (i.e., obstacles and free space). A third approach is to plan an initial path using all known information, making assumptions about parts of the environment that are unknown. As the rover acquires new information about the environment, the assumptions are updated with correct information, and the path is replanned. Generally, this approach works quite well for rover navigation, since 1) it is able to make use of continuous cost information and prior map data; 2) it works well in environments that are traversed a single time; and 3) replanning is needed only when the initial assumptions are proved invalid. This approach was adopted for the global navigation of our rover.

3 System Architecture

Our navigation system architecture is depicted in Fig. 2. The ovals are hardware, and the rectangles are modules. Each module runs as a separate process (on a separate machine if necessary), and the links shown between modules represent network messages. We use the RTC messaging system, developed at CMU, for inter-process communication. We can also run the system in simulation mode, where the only system change is that the inputs and outputs to the ovals in Fig. 2 are replaced by RTC messages to/from a graphical simulator (Section 8).

The arbiter module is responsible for combining recommendations from the local and global planners and choosing the best control action based on those

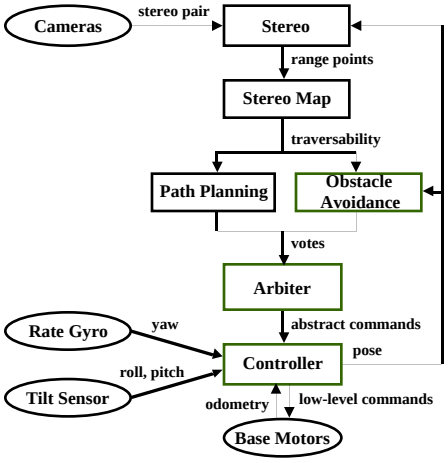

Fig. 2 Robot Architecture

recommendations. Recommendations are in the form of a set of "votes" for a fixed set of steering angles. Each voting module assigns either a veto or a vote between 0 and 1 to each command. This vote is multiplied by the module's weight in the arbiter. The command not vetoed with the highest weighted sum is sent to the controller module for execution. For this navigation system, we use a set of forward moves with different curvatures to the left and right, plus the options of turning in place, either left or right. While the arbiter can also deal with choosing vehicle speed in a similar manner, we have not yet experimented with this capability.

4 Traversability Analysis

The Stereomap module computes a traversability measure for the terrain, based on individual frames of stereo range data. The stereo vision algorithm that we use was developed at JPL and is being used on NASA's next generation Mars Rover. The output of Stereomap is a configuration-space "goodness" map that, for each grid cell, indicates how easy it would be for the rover to drive if the center of the rover passes through that cell. Stereomap essentially performs a statistical analysis of the terrain in order to handle noise in the stereo data and to facilitate the problem of fusing data from a sequence of images. It sends its output to both path planning and obstacle avoidance modules which in term combine the traversability maps over time.

The Stereomap algorithm estimates the roll, pitch and roughness of terrain patches centered at each grid cell. The terrain patches are 1.25m square, which is a conservative estimate of the rover's dimensions. The roll and pitch of a patch is estimated by using a least squares method to fit a plane to the stereo range data points that cover each rover-sized patch. The roughness is estimated as the chi-squared residual of the fit. Actually, roughness is estimated at two levels of abstraction— the residual is calculated for the complete 1.25m square patch, and also for 25cm sub-

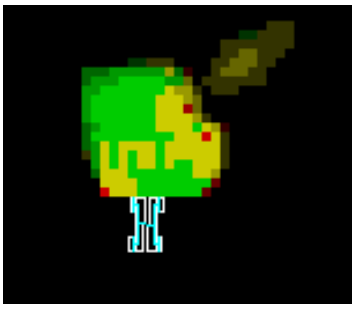

Fig. 3 Stereomap: the traversability mapper calculates the expected roll and pitch of the vehicle at each map grid cell in the sensor footprint using local least-squares plane-fitting on range data. The residual error of the plane fit is used as an estimate of the small-scale roughness of the terrain.

patches. The maximum residual over both levels is taken as the roughness measure. The roll, pitch and roughness measures are all normalized in the range [0,1], and the overall goodness of a cell is determined to be the minimum of the three measures. For efficiency in computing the plane fit and various residuals, statistics (sums and moments) are computed incrementally for each 25cm map cell, and then are combined to form larger patches.

The certainty of a terrain patch, which is a value in the range [0,1], is computed as a function of the number of points in the patch and the distribution of points. The distribution is estimated by looking at the 25cm sub-patches and determining if a significant number of these have valid data points. This prevents the algorithm from giving significant weight to patches where the data is spotty, such as appearing in only one corner of the patch. In the next section, we describe how the certainty measure is actually used.

The goodness map produced by Stereomap is always centered on the rover and, for ease of integration with the other modules, is always oriented along the axes of some arbitrary, but shared, global frame of reference. The Stereomap module publishes its map together with robot pose associated with the stereo data. These individual maps are the basic input to the local and global traversability planners, described in the following two sections.

5 Local Traversability

A crucial aspect for autonomous planetary rovers is the ability to traverse rugged terrain safely and reliably. The rover should, of course, avoid hazards such as large obstacles and depressions. However, it also should exhibit a preference for traveling on relatively flat and relatively clear terrain, all else being equal. Our navigation system accomplishes this by using a continuous traversability measure that allows the planner to reason about degrees of traversability.

Our local traversability planner (Morphin) combines

inputs from the Stereomap module (previous section) and performs a traversability analysis tuned to the capabilities of the rover. The Morphin algorithm, which is based on the Ranger algorithm of [5], has been used on a number of different robot platforms [11],[14]. Morphin combines sequences of goodness maps from the Stereomap module to form a local area map. Each time it receives a new goodness map, Morphin "ages" the existing cells in the local area map and merges the new map with the aged map. "Aging" involves multiplying the certainty of existing cells by a value (less than one) that is proportional to the distance the rover has traveled since the last map update. Cells whose certainty fall below a certain threshold are assigned zero goodness. Goodness maps are combined using a weighted average of their goodness values, with the weights proportional to the certainty values. Thus, new data is preferred, but old data still has some impact.

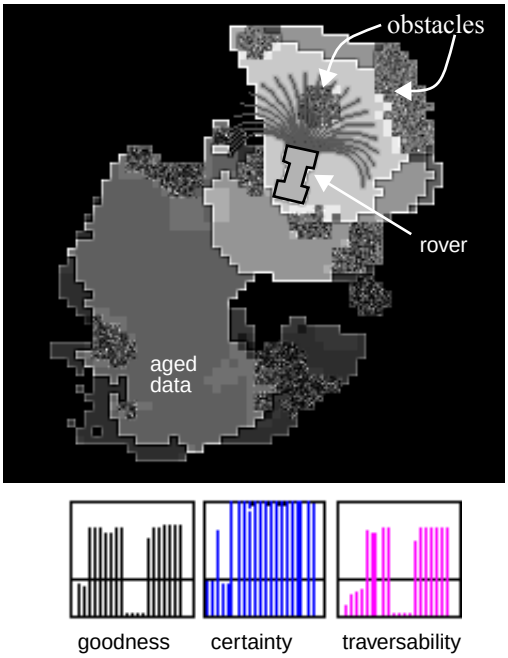

Fig. 4 (Top) A sample Morphin map with candidate rover trajectories. The darker areas indicate the areas that the rover is less certain off because they have "aged". Black areas indicate areas that have not been seen by the rover or have been completely forgotten. Obstacles are indicated by textured areas. (Bottom) The votes along each of the arcs are indicated in histograms for "goodness" and "certainty". Traversability is a weighted combination of goodness and certainty. Votes below a threshold are vetoed.

Morphin evaluates the traversability along a set of predetermined arcs. Each arc represents the path the rover would take when commanded with a particular steering angle. Morphin's vote for an arc is either the traversability value over the arc, or, a veto if the value is below a threshold. The traversability value is defined as the product of goodness and certainty. The goodness value of a cell is weighted by a function of the certainty value of the cell and the distance of the cell along the arc, while the certainty value is weighted just by the distance along the arc. Formally, the goodness of an arc, G, is given by:

$$G = \frac{\int w(s)c(s)g(s)ds}{\int w(s)c(s)ds}$$

where $w(s)$ is a piece-wise linear function that is constant for length l_u and then decreases to zero by the end of the arc at l_t. (l_t-l_u) is known as the "discounted length", $c(s)$ is the certainty measure, and $g(s)$ is the goodness measure. Similarly, the certainty of an arc, C, is given by:

$$C = \frac{\int w(s)c(s)ds}{\int w(s)ds}$$

Without discounting, obstacles at the far end of long arcs tend to affect the traversability measure along that arc, even though the rover rarely travels to the end of a given arc before turning again. Reducing the length of the arcs produces smoother trajectories, but ignores potentially useful data. Discounting parts of the arc enables driving in cluttered environments where the robot is required to get close to obstacles without running into them.

Apart from the addition of discounting, we have modified the scheme to deal with the case when all the arcs are vetoed. In other Morphin-based systems, we have used a relatively complex algorithm for backing up and turning in that case. Since Bullwinkle can turn in place, a simpler strategy was used— the local and global planners vote for two extra arcs that represent left and right point turns. Morphin always votes for these arcs with a very low value. Thus, the arcs are chosen by the arbiter only if Morphin vetoes all the forward-pointing arcs.

Since Bullwinkle has much less ground clearance than either Nomad or Ratler, it needs to detect smaller obstacles to avoid high centering. Unfortunately, these small obstacles are on the scale of stereo noise (10 cm). We are still working to address this problem, partly through parameter tweaking in the Stereomap module, and partly through investigation of an alternate method of measuring traversability using a multiresolution technique based on the Laplacian pyramid[3].

6 Global Planning

Our approach is to use D* (Dynamic A*), which allows replanning to occur incrementally and optimally in real-time [15][16]. Like A*, D* plans an initial path from the robot's start state to the goal state using all available information, known or assumed. As the rover follows the path, its sensors can discover discrepancies between the map and world. The map is updated, possibly invalidating the optimality or feasibility of the current path. D* uses incremental graph theory techniques to compute a new, optimal path to the goal, or to re-validate the old one. D* accomplishes this by computing, saving, and re-using partial

solutions to the planning problem. In most cases, D* can recompute the optimal path quickly, by calculating a "local patch" to the existing path that moves the rover around a small obstacle. In other cases, more extensive re-planning is required to compute entirely new solutions. Either way, the D* algorithm performs close to the minimum computation necessary to re-plan the path to the goal. For large environments, D* has been shown to be hundreds of times faster than re-planning from scratch using A*.

D* is optimal in the following sense. For every point P along its traverse, the rover follows an optimal path to the goal assuming the correctness of all map information acquired in aggregate to point P. In general, traverses produced by D* have been shown empirically to be low in cost. If the unknown parts of the world are assumed to be obstacle-free, then D* is complete in the sense that it will reach the goal if a path exists.

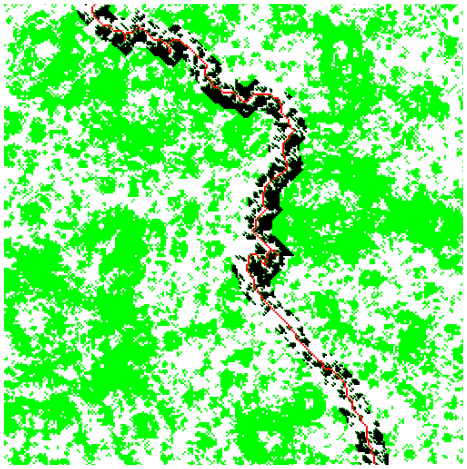

Fig. 5 Traverse in a dense environment starting with no prior knowledge of the word. The dark area shows the obstacles that are discovered as the robot moves through the world.

Large outdoor areas also likely contain sparsely or fractally scattered obstacles, leaving large empty space among obstacles. Methods that use uniform grid representations must allocate large amounts of memory for regions that may never be traversed, or contain any obstacles. Efficiency in map representation can be obtained by the use of quadtrees, but at a cost of optimality. Recently, a new data structure called a *framed quadtree* has been suggested as means to overcome some of the issues related to the use of quadtrees. We have used this data structure to extend the D* search algorithm that has hitherto used uniform (regular) grid cells to represent terrain. In many cases, usage of this representation can produce, shorter, straighter paths at a lower computational cost. An example path found in a cluttered environment and the spatial representation of the terrain is shown in Fig. 6. A more detailed discussion of these issues can be found in [18][19].

7 Combining local and global planning

The global planner assigns low cost to unknown areas

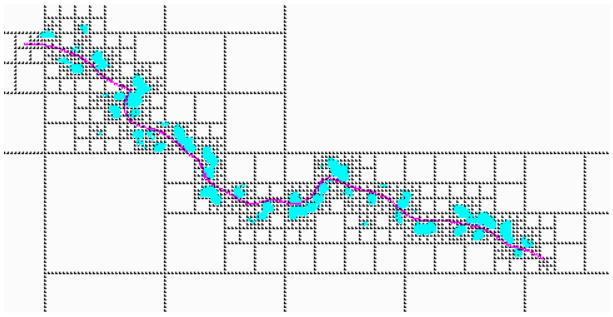

Fig. 6 The use of *framed quadtrees* to represent space provides an efficiency in memory and computational resources when used with D*. The tessellation of space illustrates the structure developed by framed quadtrees.

(which the robot is expected to see before actually driving into), while local obstacle avoidance keeps the rover safe by vetoing arcs which lead immediately into the unknown. Optimizing the ratio of weights to be assigned to local vs. global planning is a subject of ongoing research. To be conservative, local planning is weighed higher than global planning, at the cost of path optimality. An issue that we have not answered decisively is how to define the semantics of the different levels of voting.

Another question relates to the scale chosen for local obstacle avoidance. In addition to keeping the rover safe from immediate obstacles by vetoing, the local module can coach the rover away from more distant obstacles by adjusting the strength of its votes. However, at larger scales, the global knowledge advantage of the global module outweighs the better reaction time and finer map resolution advantage of the local module. Our experiments with a 1 m scale rover suggest that, for this reason, the maximum useful length of steering arcs considered by the local module is around 3-5 m. There are also many possible schemes for discounting the weight of obstacles' effect on the goodness value of an arc in proportion to their distance along the arc. Current experiments will tell us how to improve rover efficiency by tuning the arc lengths and discounting scheme.

8 Results

Our system has driven Bullwinkle 100m in outdoor environments as shown in Fig. 7 at an average speed of 15 cm/sec. The environments we have tackled with Bullwinkle to date consist mainly of free space (although the free space is not necessarily level) and clear two dimensional obstacles. In the near future we expect to navigate in environments populated by small obstacles, some of which will be traversable. However, we have conducted experiments with cluttered environments in simulation. Our simulator models a rover navigating in a cluttered environment as shown in Fig. 8. Apart from simulating the rover kinematics, the simulator also models the stereo vision system, providing

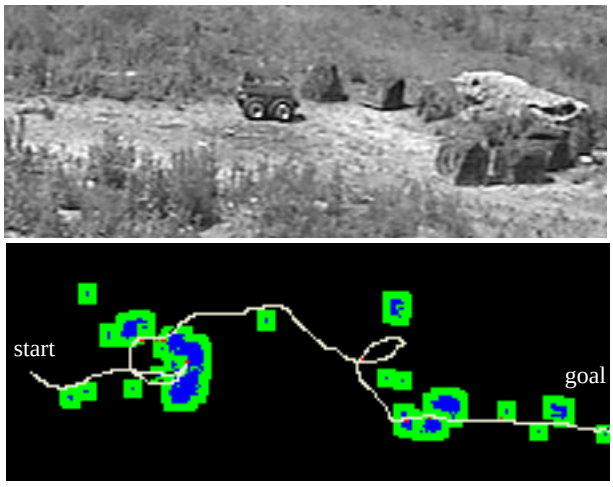

Fig. 7 Results from a field trial (above) Bullwinkle negotiates a cul-de-sac with no prior knowledge of the environment. (below) Final map created by D* after a 60 m traverse.

synthetic range data in exactly the same way as produced by our stereo vision algorithms. In fact, the simulator is attached to the rest of the system seamlessly; the other modules are not aware that they are driving the real robot or the simulated one.

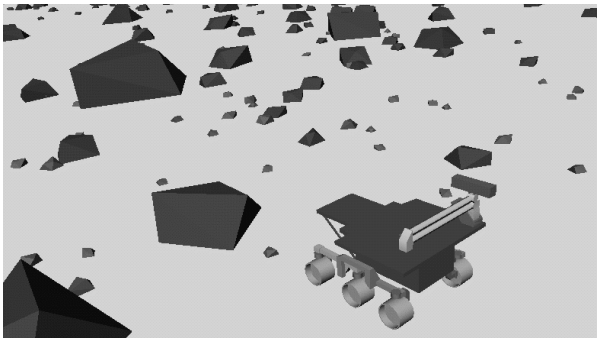

Fig. 8 Simulated rover navigating in a cluttered environment.

We have conducted a large set of experiments in simulation aimed at understanding the effect of adjusting control parameters. For example, Fig. 9 shows two simulated runs in the same environment where the ratio of weights given to D* and Morphin were changed.

TABLE I tabulates the results from 240 simulation experiments. The simulated rover moves at a speed of 5 cm/sec while our simulator runs at twice as fast as real time. All modules (Stereomap, Morphin, D* and Arbiter) were run on one SGI R10000 processor. Notes:

- The control parameter set weighs Morphin twice as high as D*. The arcs are 3 m long and the last 2m's are discounted.
- Time taken to reach the goal is a function of the distance travelled as well as the number of point turns.
- Body collisions are those collisions where the body of the rover collided with objects above 27 cm while wheel collisions are those collisions in which the wheels collided with objects that are at least 20 cm high.

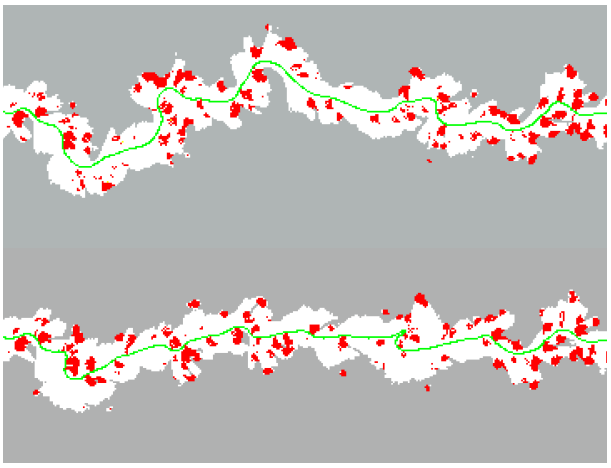

Fig. 9 Two simulated runs in the same environment showing the difference performance when the relative weights between D* and Morphin are changed. The distance between start and goal is 100m. The large grey areas represent regions that remain unexplored. (Top) Morphin is weighted 5 times more than D*; distance traveled = 123.8m; 1 point turn was required (Bottom) D* is weighted 2 times more than Morphin; distance travelled = 114.9m; 4 point turns were required.

Ratio of D* to Morphin weights	**Time to goal reach goal (s)**	**Distance travelled (m)**	**Body Collisions**	**Wheel Collisions**	**# of point turns**
0.2	2569/355	122.0/14.26	0.1/0.3	2.4/1.9	3.1/2.7
Control (0.5)	2401/276	114.8/10.23	0.1/0.3	3.0/2.0	3.1/2.6
1	2436/337	113.45/9.25	0.4/0.6	3.9/2.5	4.4/4.4
2	2420/344	111.82/8.63	0.4/0.8	4.0/2.1	4.9/3.8
10	2500/403	113.56/10.9	0.4/0.9	4.1/2.8	6.7/4.4
Arc length (l_f)/ discounted length (l_f-l_u)(m)	**Time to goal reach goal (s)**	**Distance travelled (m)**	**Body Collisions**	**Wheel Collisions**	**# of point turns**
1/0	2884/428	126.03/12.8	0.7/0.8	6.7/3.2	10.4/4.8
2/0	2496/400	115.63/11.29	0.3/0.5	3.2/2.3	6.2/5.5
3/0	2692/449	122.27/14.4	0.2/0.4	2.8/1.4	11.6/7.7
2/1	2534/377	117.0/12.15	0.4/0.6	3.6/2.1	4.8/3.5
3/1	2614/418	119.36/9.92	0.3/0.7	3.1/2.4	8.2/6.3
Control (3/2)	2401/276	114.8/10.23	0.1/0.3	3.0/2.0	3.1/2.6

TABLE I The effect of adjusting key control parameters on performance metrics. Each cell contains the mean and standard deviation for each run. Each control parameter set was used in 30 runs in which the distance between the start and goal was100 m. The upper part of the table shows performance for varying weights between D* and Morphin. The lower part of the table refers to adjustments of arc lengths and the discounting scheme. For example, the second row labeled "2/1" refers to arc lengths that are 2 m long and the last 1m is discounted. Note that of a total of 330 runs, the rover didn't reach the goal in only one case. We suspect that this was due to a simulator error.

Analysis of the results indicates that performance is not very sensitive to change in parameter settings. A few

trends are noticeable, however:

- *The greater the weight given to D*, the closer the rover gets to the obstacles*. This often results in shorter paths but the closer the robot gets to obstacles, the larger the chance that the robot will collide with obstacles and will need to point turn. Conversely, if D* is weighted much less than Morphin (less than 5 times), the robot is sometimes not able to reach the goal, especially in very cluttered environments.
- *Discounting arcs is useful*. If the arcs are not discounted, the robot is more conservative. While it has fewer collisions, the rover shies away from obstacles from a distance. For example, in cases where a path between obstacles would require an S-curve, the circular arcs used to represent future trajectories can lead to the conclusion that no path exists in between the obstacles resulting a large number of point turns. Discounting lets the robot approach obstacles and find paths in between them.

9 Future Work

In the near future, we expect to perform the same type of parameter tests on our robot testbed as were performed in simulation. We will also experiment with our rover in more complex terrains. This system will also be ported to a NASA rover in the near future.

Acknowledgments

The authors would like to thank Bruce Digney, Stewart Moorehead and Greg Armstrong for their help with the testbed vehicle. This research was sponsored in part by NASA, by the Intelligent Robotics program under contract 99-614.

References

[1] Ronald C. Arkin, “Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior”, in Proc. Intnl Conf on Robotics and Automation, Raleigh, NC, March, 1987.

[2] Johann Borenstein and Yoram Koren, “The Vector Field Histogram -- Fast Obstacle Avoidance for Mobile Robots”, IEEE Trans. on Robotics and Automation, 7:3, pp. 278-288, 1991.

[3] Peter Burt and Edward Adelson, “The Laplacian Pyramid as a Compact Image Code,” IEEE Transactions on Communications, 31:4, pp. 532-540, 1983.

[4] Hebert, Martial H., “SMARTY: Point-Based Range Processing for Autonomous Driving,” *Intelligent Unmanned Ground Vehicle*, Martial H. Hebert, Charles Thorpe, and Anthony Stentz, editors, Kluwer Academic Publishers, 1997.

[5] Kelly, A, “An Intelligent Predictive Control Approach to the High Speed Cross Country Autonomous Navigation Problem,” Ph.D Thesis, 1995, Carnegie Mellon University, Pittsburgh, PA 15213.

[6] Donald Gennery, “Traversability Analysis and Path Planning for a Planetary Rover”, Autonomous Robots, 1999.

[7] Korf, R.E., “Real-Time Heuristic Search: First Results,” Proc of the Sixth National Conf on Artif. Intelligence, July 1987.

[8] Laubach, S., Burdick, J., Matthies, L., “Autonomous Path-Planning for the Rocky 7 Prototype Microrover,” in Proc. IEEE Intn’l Conf on Robotics and Automation, 1998.

[9] Lumelsky, V.J., Stepanov, A.A., “Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment”, IEEE Transactions on Automatic Control, Vol. AC-31, No. 11, November 1986.

[10] Larry Matthies, Erann Gat, R. Harrison, Brian Wilcox, Rich Volpe, Todd Litwin, “Mars Microrover Navigation: Performance Evaluation and Enhancement”, Autonomous Robots, 2:291-311, 1995.

[11] S.J. Moorehead, R. Simmons, D. Apostolopoulos, W. Whittaker, Autonomous Navigation Field Results of a Planetary Analog Robot in Antarctica, Intn’l Symposium on Artificial Intelligence, Robotics and Automation in Space, 1999.

[12] Pirzadeh, A., Snyder, W., “A Unified Solution to Coverage and Search in Explored and Unexplored Terrains Using Indirect Control”, in Proc. IEEE Intn’l Conf on Robotics and Automation, 1990.

[13] H. Seraji, “Traversability Index: A new concept for planetary rovers,” in Proc. IEEE Intn’l Conf on Robotics and Automation.

[14] R. Simmons, E. Krotkov, L. Chrisman, F. Cozman, R. Goodwin, M. Hebert, L. Katragadda, S. Koenig, G. Krishnaswamy, Y. Shinoda, W. Whittaker, and P. Klarer. “Experience with Rover Navigation for Lunar-Like Terrains,” in Proc. Conf on Intelligent Robots and Systems (IROS), Pittsburgh PA, 1995.

[15] A. Stentz, Optimal and efficient path planning for partially-known environments, in Proc. IEEE Intn’l Conf on Robotics and Automation, May 1994.

[16] Stentz, A., “Best Information Planning for Unknown, Uncertain, and Changing Domains,” AAAI-97 Workshop on On-line-Search.

[17] Stentz A., Hebert, M., “A Complete Navigation System for Goal Acquisition in Unknown Environments,” Autonomous Robots, 2(2), 1995.

[18] A. Yahja, A. Stentz, S. Singh, and B. Brumitt, Framed-quadtree path planning for mobile robots operating in sparse environments, in: Proc. IEEE Intn’l Conf on Robotics and Automation, (ICRA), Leuven, Belgium, May 1998.

[19] A. Yahja, Sanjiv Singh and Anthony Stentz, An efficient on-line path planner for outdoor mobile robots, To appear in *Robotics and Autonomous System* s journal, Elsevier Science.

[20] Thrun, S., “An Approach to Learning Mobile Robot Navigation”, Robotics and Autonomous Systems 15(1995): 301-19.