Global Planning on the Mars Exploration Rovers: Software Integration and Surface Testing

Joseph Carsten and Arturo Rankin
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109
e-mail: joseph.carsten@jpl.nasa.gov, arturo.rankin@jpl.nasa.gov

Dave Ferguson
Intel Research Pittsburgh
Pittsburgh, Pennsylvania 15213
e-mail: dave.ferguson@intel.com

Anthony Stentz
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
e-mail: tony@cmu.edu

Received 2 July 2008; accepted 6 January 2009

In January 2004, NASA's twin Mars Exploration Rovers (MERs), Spirit and Opportunity, began searching the surface of Mars for evidence of past water activity. To localize and approach scientifically interesting targets, the rovers employ an onboard navigation system. Given the latency in sending commands from Earth to the Martian rovers (and in receiving return data), a high level of navigational autonomy is desirable. Autonomous navigation with hazard avoidance (AutoNav) is currently performed using a local path planner called GESTALT (grid-based estimation of surface traversability applied to local terrain) that incorporates terrain and obstacle information generated from stereo cameras. GESTALT works well at guiding the rovers around narrow and isolated hazards; however, it is susceptible to failure when clusters of closely spaced, nontraversable rocks form extended obstacles. In May 2005, a new technology task was initiated at the Jet Propulsion Laboratory to address this limitation. Specifically, a version of the Field D* global path planner was integrated into MER flight software, enabling simultaneous local and global planning during AutoNav. A revised version of AutoNav was then uploaded to the rovers during the summer of 2006. In this paper we describe how this

Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/rob.20287

integration of global planning into the MER flight software was performed and provide results from both the MER surface system test bed rover and five fully autonomous runs by Opportunity on Mars.

1. INTRODUCTION

In January 2004, two robotic vehicles landed on Mars as part of NASA's Mars Exploration Rover (MER) mission (see Figure 1). Since that time, these two rovers, Spirit (Leger et al., 2005) and Opportunity (Biesiadecki et al., 2005), have been searching the Martian surface for evidence of past water activity. Directing rover activities poses an interesting challenge for scientists and engineers. It can take as long as 26 min for a signal from Earth to reach Mars (and vice versa). This makes teleoperation of the rovers infeasible. In addition, line-of-sight and power constraints further complicate the situation. To overcome these factors, each rover is sent a sequence of commands at the beginning of each Martian day (sol). This command sequence lays out all activities to be performed by the rover during the sol. The rover then executes the command sequence without any human intervention. In general, before the rover enters into a sleep mode for the night, it will send data back to Earth. These data are then used to plan activities for the following sol. Owing to the fact that commands are received only once per sol, rover autonomy is critical. The more autonomous the rover is, the more activities it can accomplish each sol. Here, we will focus our attention on the navigation system, but this observation applies to all rover behaviors.

Figure 1. Artist's rendition of a MER. Courtesy NASA/JPL-Caltech.

The purpose of the navigation system is to move the rover around the Martian surface in order to locate and approach scientifically interesting targets. To begin the process, engineers on Earth identify a goal location that they would like the rover to reach. Typically, images returned by the rover are used to select this goal. Two main methods can be used to reach this goal. The first and simplest is the blind drive. During a blind drive, the rover does not attempt to identify hazardous terrain and simply drives toward the goal location. The second option is autonomous navigation with hazard avoidance (AutoNav). In this case, the rover autonomously identifies hazards, such as large rocks, and steers around them on its way to the goal.

There are advantages and disadvantages to each approach. During a blind drive, the rover can cover a larger distance in a given time period because it does not have to process imagery of the surrounding terrain. However, this means that the engineers on Earth must verify that the terrain between the rover and the goal is free from hazards before commanding the drive. On the other hand, AutoNav is slower but can keep the rover safe even in regions unseen by engineers on Earth. Often, the two methods are utilized in tandem. First a blind drive is commanded as far out as engineers can be sure of safety. Then AutoNav is used to make additional progress through unknown terrain. Thus, the increased autonomy provided by AutoNav allows much more forward progress to be made during a sol.

Although AutoNav is usually able to guide the rover to the goal, there are known circumstances under which it is susceptible to failure and the rover does not reach the goal. In July 2006, a new version of the MER flight software was successfully uploaded to the rovers. Owing to the complexity and number of changes, a software patch was infeasible and a full flight software load was necessary (Greco & Snyder, 2005). In addition to bug fixes and other improvements, four new technologies were included. These new technologies were visual target tracking, onboard dust devil and cloud detection, autonomous placement of the instrument deployment device, and a global path planner designed to overcome some of the shortcomings of AutoNav (Schenker, 2006).

A description of this planner, its integration into the flight software, test results on Earth, and test results on Mars are described in the remainder of this paper.

2. AUTONAV

2.1. Overview

The purpose of AutoNav is to enable the rover to safely traverse unknown terrain. AutoNav is based on the GESTALT (grid-based estimation of surface traversability applied to local terrain) algorithm (Biesiadecki & Maimone, 2006; Goldberg, Maimone, & Matthies, 2002). AutoNav uses stereo image pairs captured by the rover's onboard camera system to gather geometric information about the surrounding terrain. These images are processed to create a model of the local terrain. Part of this model is a goodness map. This goodness map is grid based and represents an overhead view of a model of the terrain. Each grid cell in the map contains a goodness value. High goodness values indicate easily traversable terrain, and low goodness values indicate hazardous areas. The map is constructed in configuration space, meaning that hazards are expanded by the rover radius in all directions before their representations are included in the map. This allows the rover to be treated as a point in future computations.

Once the terrain has been evaluated, a set of candidate arcs (short paths from the current rover location) is considered. Nominally, the arc set consists of forward and backward arcs of varying curvature, as well as point turns to a variety of headings. Each arc is evaluated based on three criteria: avoiding hazards, minimizing steering time, and reaching the goal. For each arc, a vote based on each of these criteria is generated. The goodness map is used to generate the hazard avoidance vote. Arcs that travel through cells that are difficult or dangerous to traverse receive low votes. Steering bias votes are constructed based on the amount of time that is needed to turn the wheels from the current heading to the heading required to execute the candidate arc. Arcs requiring less steering time receive higher votes. Waypoint votes are constructed based on the final criterion: reaching the goal. Arcs that move the rover closer to the goal location receive higher waypoint votes. The three votes are then weighted and merged to generate a final vote for each arc. Once votes have been generated, the best arc is selected for execution. The rover then drives a short, predetermined distance along the selected arc. This process is repeated (evaluate terrain, select arc, drive) until the goal is reached, a prescribed timeout period expires, or a fault is encountered.

2.2. Shortcomings

AutoNav is very good at keeping the rover safe and usually gets the rover to the goal location. However, in some instances AutoNav is not able to reach the goal. Figure 2 illustrates one such situation. In this case, Spirit spent approximately 105 min and 47 drive steps trying to get around a cluster of rocks but was unable to autonomously do so. The cause of this failure was the simple method used to construct waypoint votes by AutoNav. Arcs that decrease the Euclidean distance between the rover and the goal always receive higher votes. This biases the rover to take straight-line paths to its goals. When the waypoint votes are then merged with hazard avoidance votes, some deviation to get around small hazards is possible. However, the amount of deviation that can occur is fairly minimal. When the rover encounters a large hazard in its path, the waypoint votes and hazard avoidance votes conflict severely. The hazard avoidance votes will not allow the rover to drive

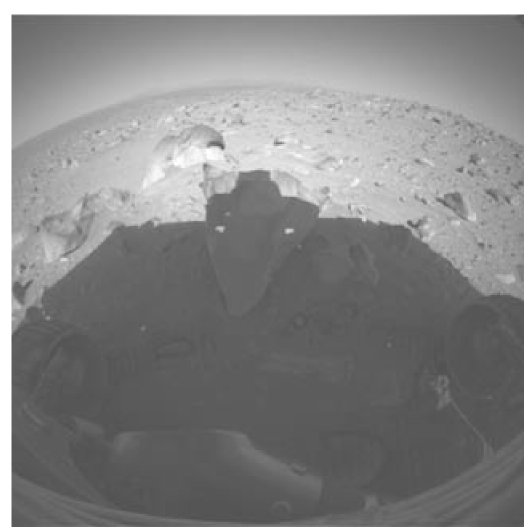


Figure 2. On sol 108, Spirit was unable to autonomously navigate to a goal location on the other side of this cluster of rocks. This image was captured by one of the front hazard avoidance cameras mounted on the body of the rover. A portion of the scene is in a shadow cast by the rover. Courtesy NASA/JPL-Caltech.

through the unsafe area, and the waypoint votes will not allow enough deviation from the straight-line path for the rover to get around the hazard. As a result, the rover becomes stuck and is unable to reach the goal.

3. GLOBAL PATH PLANNING

For improved performance, a better waypoint vote metric is needed; something that is more accurate than Euclidean distance. A better metric can be produced by planning paths to the goal that take into account all of the obstacles in the environment. Typically, the environment will be only partially known to the rover, and thus complete information regarding the obstacles will not be available. However, incorporating obstacle information that *is* available into these global plans typically provides much better estimates than Euclidean distance, and these estimates only improve in accuracy as more information is acquired during the rover's traverse. Such an approach has been widely used in outdoor mobile robot navigation (Brock & Khatib; 1999; Kelly, 1995; Kelly et al., 2006; Kolski, Ferguson, Bellino, & Siegwart, 2006; Singh et al., 2000; Stentz & Hebert, 1995).

We have extended the AutoNav system to use the Field D* algorithm to generate these global paths. Field D* is a planning algorithm that uses interpolation to provide direct, low-cost paths through two-dimensional, grid-based representations of an environment (Ferguson & Stentz, 2006). Each grid cell is assigned a cost of traversal. Based on these costs, the algorithm generates a path between two locations, with the aim of minimizing the cost of traversing that path.

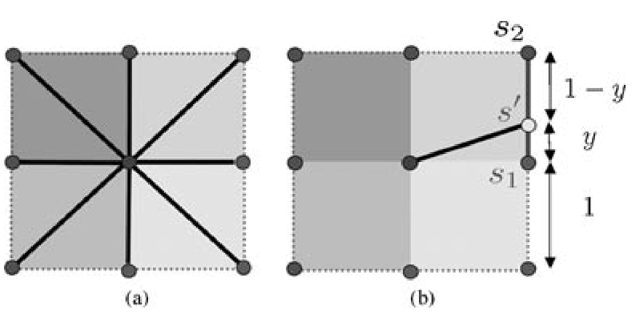


Figure 3. (a) The typical transitions (in black) allowed from a node (shown at the center) in a uniform grid. Notice that only headings of 45-deg increments are available. (b) Using linear interpolation, the path cost of any point s' on an edge between two grid nodes s_1 and s_2 can be approximated. This can be used to plan paths through grids that are not restricted to just the 45-deg heading transitions. Figure reproduced from Ferguson and Stentz (2006).

Although two-dimensional grids present an easy and computationally efficient way to represent the environment, a major limitation of classic grid-based planning algorithms is the restricted nature of the paths produced. For example, classic grid-based planners usually restrict paths to transitioning between adjacent grid cell centers or corners, resulting in paths that are suboptimal in length and involve unnecessary turning. Figure 3(a) shows the typical transitions allowed from a particular grid cell.

The Field D* algorithm removes this restriction and allows paths to transition through any point on a neighboring grid cell *edge*, rather than just the neighboring grid cell corners or centers. To do this efficiently, it uses linear interpolation to approximate the path cost to any point along a grid cell edge, given the path costs to the endpoints. Equation (1) and Figure 3(b) illustrate how linear interpolation is used to provide an estimate of the path cost to an edge node s' given the path costs to end nodes s_1 and s_2. Here, y is the distance between s_1 and s', measured as a fraction of the length of a grid cell side:

$$\text{PathCost}(s') \approx y \cdot \text{PathCost}(s_2) + (1-y) \cdot \text{PathCost}(s_1).$$

As a result, Field D* is able to provide much more direct, less-costly paths than standard grid-based planners without sacrificing real-time performance. It is also able to efficiently repair its solutions as new information is received, for example, through onboard sensors. Figure 4 shows a path planned by Field D* along with a classic grid-based path.

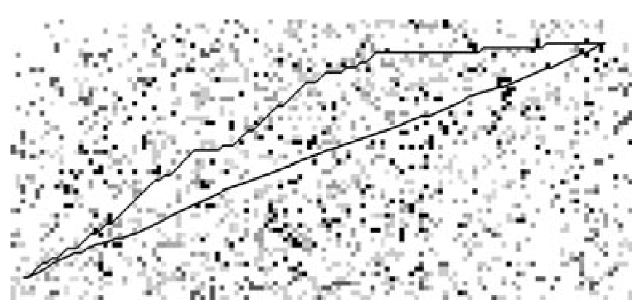


Figure 4. Paths produced by classic grid-based planners (uppermost path) and Field D* (lower path) in a 150 × 60 uniform resolution grid. Darker cells represent higher cost areas. Figure reproduced from Ferguson and Stentz (2006).

4. INTEGRATION

At the highest level, using Field D* to improve AutoNav involves two main tasks. The first is providing terrain information to Field D* in a form it can utilize. The second is using Field D* to generate steering recommendations in a form that AutoNav can understand.

4.1. Cost Map

Field D* uses a uniform grid as the basis of its world model. Each grid cell contains a value that represents the cost of traversing the width of the cell. Fortuitously, this is very similar to the goodness map representation of the world maintained by AutoNav. However, the goodness map is always centered on the rover location and stores only information about the local terrain. Field D* plans on a global scale and must therefore store a much larger map. In addition, the Field D* map is fixed to the environment and does not move along with the rover. There are also several other key differences between the two representations. Field D* operates on cost values, where more easily traversable terrain has a lower cost, but AutoNav stores a goodness map, where more easily traversable terrain has a higher goodness. In addition, grid cells in the goodness map can have "unknown" goodness. This indicates that there is not enough information about that cell location to determine its traversability. By default, the Field D* cost map has no such value.

Using the goodness map to update the cost map is fairly straightforward. First, because there is no notion of unknown cost, the entire cost map must be initialized to a given cost value. Initializing all cells to a low cost means the rover will be much more inclined to explore unseen regions. On the other hand, initializing to a high cost means that the rover will prefer to stay in regions it has already seen. Here, a midrange cost value was chosen. Next, at each step of the traverse, the position of the goodness map inside the larger cost map is determined. Then each goodness cell that is not unknown is merely translated into a cost value and placed into the corresponding cost grid cell. For this to operate correctly, the goodness grid cells and cost grid cells must be the same size. In addition, grid cell boundaries in the goodness map must align with those in the cost map. These issues are addressed when the maps are created. Each goodness value is translated into a cost value as follows. Cells with very low goodness are set to a special cost value representing obstacle. Field D* will not plan paths through these cells. All other goodness values are inverted and then scaled to the range of cost values to produce corresponding costs. By virtue of its much larger map, Field D* tracks everything the rover has seen, even when it has been long forgotten by the local goodness map. Figure 5 shows a

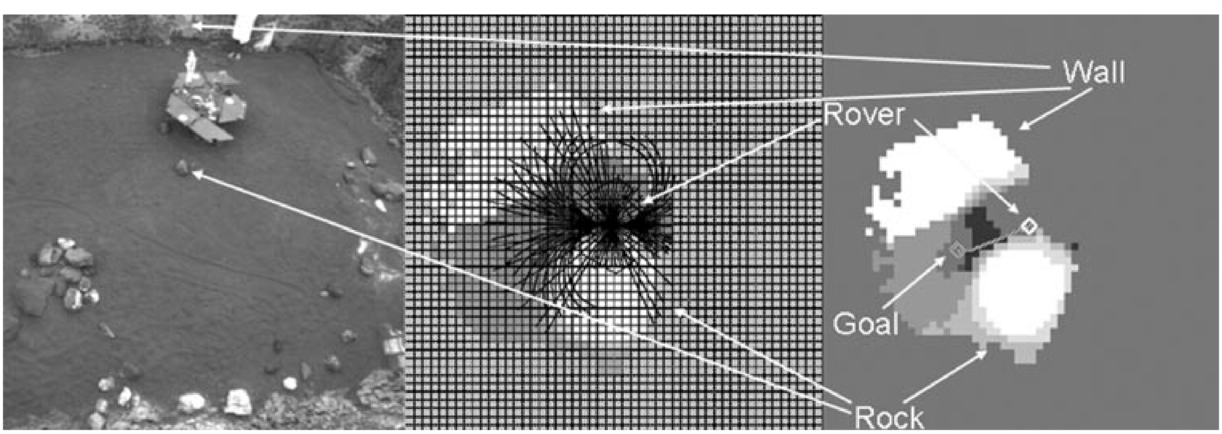


Figure 5. The left-hand image is an overhead view of a test bed rover in an indoor sandbox. The middle image is the corresponding goodness map, and the Field D* cost map is shown in the right-hand image. Checkered cells around the perimeter of the goodness map have unknown traversability. All other cells are colored based on a gradient between black (high goodness/low cost) and white (low goodness/high cost). The candidate arc set is shown in black in the goodness map. Note that the entire goodness map is presented, but only a small portion of the cost map is shown in here. Also note that the maps are constructed in configuration space, in a local frame not aligned with the sandbox walls.

goodness map and the corresponding portion of the Field D* cost map.

4.2. Votes

Once the cost map has been populated, a method is needed to use Field D* to influence arc selection. The output of Field D* is the cost of traversing the optimal path from any query point to the goal location. However, the easiest way to provide steering recommendations to the rest of the system is through arc votes. Therefore, an approach for converting costs of traversal to arc votes is necessary.

To begin this process, Field D* is used to compute the cost of traversal from the end of each candidate arc to the goal. Taken individually, these traversal costs mean very little. If the rover is 50 m from the goal, the traversal cost for a given arc will be much higher than if the rover is 10 m from the goal. This is merely due to the fact that there is much more ground to cover in the first case. Fundamentally, arc votes are just a way of ranking the candidate arcs from best to worst. When taken relative to each other, the traversal costs provide a similar ranking mechanism.

The arc with the lowest cost of traversal to the goal is the best, and the arc with the highest cost is the worst. Numerical vote values are assigned using a weighted sum of v_{scale} and v_{close}, which are given in Eqs. (1) and (2). v_{max} is the maximum possible vote, c_{max} and c_{min} are the maximum and minimum traversal costs computed during the current arc set evaluation, and c_i is the traversal cost for a given arc i. The minimum vote value is zero. v_{max} serves as a way to adjust the weight of the Field D* votes relative to the other voting modules (hazard avoidance and steering bias). The larger the value of v_{max}, the more influence Field D* votes have on the final path selected:

$$v_{\text{scale}_i} = v_{\text{max}} * (c_{\text{max}} - c_i)/(c_{\text{max}} - c_{\text{min}}), \quad (1)$$

$$v_{\text{close}_i} = v_{\text{max}} * c_{\text{min}}/c_i. \quad (2)$$

v_{scale} is a standard linear scaling of the cost values into vote values. v_{close} bases vote values on how close the rover is to the goal. The closer the rover is to the goal, the greater the range of vote values that is generated. When the rover is far from the goal, $c_{\text{min}}/c_{\text{max}}$ will be close to one and all votes will be close to v_{max}. On the other hand, when the rover is close to the goal, $c_{\text{min}}/c_{\text{max}}$ will be close to zero, and the votes will be spread from zero to v_{max}. Alone, v_{close} is not particularly useful (especially when the rover is far from the goal), but when combined with v_{scale} it can be helpful. When combined with v_{scale}, v_{close} serves to reduce the range of vote values when the rover is far from the goal. This means that the preference for one arc over another is less pronounced. When the rover is far from the goal, it is not critical exactly which arc is taken (as long as the rover is moving in generally the right direction). In this case, it may be advantageous to let the other voting modules (steering bias and hazard avoidance) have more influence over the final arc selection. However, as the rover gets closer to the goal, the exact arc selected is more important and thus the entire range of vote values is utilized. Generally when combining the two vote values, v_{scale} receives a significantly higher weight than v_{close}.

Once these votes have been constructed, they replace the waypoint votes constructed by GESTALT. They are then combined with steering bias and hazard avoidance votes in order to select the arc that will be followed. When constructing Field D* votes, it is possible that several arcs may have identical costs of traversal. Nothing special need occur to handle this situation. These arcs are merely assigned equal Field D* vote values. The GESTALT arc selection algorithm handles combining these votes with steering bias and hazard avoidance votes, as well as breaking any ties that might occur in the final combined vote values (Goldberg et al., 2002). Once the naive waypoint votes are replaced with those generated using Field D*, the AutoNav system becomes much more robust.

4.3. Limitations

It should be noted that AutoNav (both with and without Field D*) assumes that the rover position is known and that candidate arcs can be executed nominally. There are cases in which these assumptions are violated. For instance, on sandy slopes the wheels may slip significantly, causing the estimated rover position to be erroneous. In addition, mechanical failure of wheel actuators can cause arcs to be executed abnormally. In these cases, AutoNav performance may be degraded. Although AutoNav makes no attempt to directly address these issues, other technologies can often be used to overcome them. For instance, visual odometery can be used in conjunction with AutoNav to maintain an accurate estimate of rover position, regardless of wheel slip (Cheng, Maimone, & Matthies, 2005).

5. RESOURCE LIMITATIONS

The Mars rovers are constrained by very limited computational resources. The onboard computer uses a radiation-hardened RAD6K processor running at 20 MHz and has 128 Mbytes of DRAM (Biesiadecki & Maimone, 2006). To make matters worse, these already limited resources must be shared among the 97 tasks (including AutoNav) that make up the onboard flight software (Reeves & Snyder, 2005). In light of these constraints, optimizations were made to the Field D* algorithm to improve efficiency. Specifically, the path cost minimization step of the algorithm is precomputed, and the results are stored in a lookup table that is accessed at run time. This significantly decreases the computation time required for planning. See Ferguson and Stentz (2006) for more details on how this is performed.

Another constraint is the limited bandwidth available to send data back to Earth. The data can be grouped into two broad categories: engineering data and science data. Science data contain information about Mars that is of interest to scientists. Engineering data are used to monitor the status of the rover and contain information that is useful should an anomaly occur. Telemetry generated by Field D* falls into this category. Because the main purpose of the mission is to better understand Mars, it is desirable to limit the engineering data to a minimum in order to maximize the amount of science data that can be downlinked.

In the case of Field D*, CPU utilization, memory usage, and telemetry volume are all tied to the size of the Field D* cost map. Larger maps mean more resource usage. Therefore, it is advantageous to find ways to reduce the map size while still obtaining good path planning results. Each of the three optimization techniques described below (automatic map recentering, coarse resolution maps, and map filtering) were implemented and are utilized during AutoNav.

5.1. Automatic Recentering

For Field D* to plan a path between a start location and a goal, both the start location and goal location must be located within the cost map. Therefore, the farther the rover is from the selected goal, the larger the map must be. To allow long traverses that would be infeasible due to memory constraints, a scheme was developed to overcome this limitation. The constraint that the user-selected goal must be within the bounds of the cost map is lifted, and any arbitrary goal location is allowed. If the goal happens to be outside of the cost map, an intermediate goal is selected that resides on the boundary of the cost map. The intermediate goal is placed at the point where the straight line between the current rover location and user-selected goal intersects the boundary of the cost map. However, this does not completely solve the problem. Now the rover is being guided to a point on the edge of the map and not the user-selected goal. For the rover to reach the user-selected goal, map recentering is needed. During the map update phase, if any portion of the local goodness map falls outside the cost map, the cost map is recentered on the current rover position.

Recentering does not alter any memory allocations but instead merely adjusts the world coordinates of the map center. To make the map consistent with the new coordinates, grid cells that are common to both the old map and new map are copied across the map to their new location. All other areas are cleared to the nominal cost value. Once this is done, a new goal is placed within the cost map. If the user-selected goal is within the new map bounds, the goal is placed there. If not, another intermediate goal is placed using the procedure outlined earlier. This recentering and intermediate goal placement is repeated until the user-selected goal is reached. Figure 6 illustrates the recentering process.

This approach has some limitations. There is a performance penalty whenever the map is recentered. Field D* is efficient because it does not have to replan from scratch when new costs are discovered. Instead, it is able to reuse the results of previous planning and repair the needed paths. However, because Field D* begins its search at the goal, whenever the goal is moved, all planning information is reset and the next path must be planned from scratch. To minimize this effect, the intermediate goal is not updated every time the rover moves. Instead, a new goal is placed only when the map is recentered. Map recentering is an infrequent event, and therefore the overall performance impact is minor.

There is another limitation to this approach. It is possible that the intermediate goal could be placed inside an obstacle. When the goal is inside an obstacle, Field D* is unable to plan any paths and will fail. However, this problem is unlikely to be encountered in practice. Usually, the intermediate goal is ahead of the rover, in an area not yet visited. Because the rover

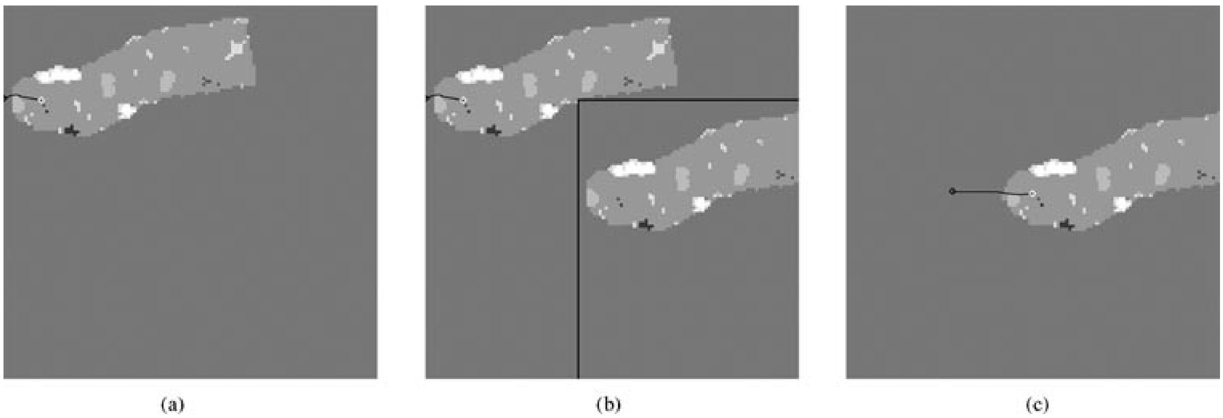


Figure 6. Field D* cost map and the recentering process. The rover is represented by a white diamond, and the goal is shown as a black diamond. Recentering is needed for the map shown in (a). Cells common to the old and new map are copied to their new location in the lower right of (b). Shown in (c) is the final map after cells not common to both the old and new maps have been cleared, the cost map has been updated from the most recent goodness map, and the goal location has been recalculated.

has not seen the terrain around the intermediate goal, that location cannot be obstacle in the cost map until the rover is close enough to evaluate that terrain. The map is recentered slightly before the rover can see the edge of the map. Therefore, in general, the rover has never evaluated the terrain under any current intermediate goal. The exception is if the rover is in the process of backtracking a significant distance in order to navigate around a very large hazard. In this case, the ultimate goal is behind the rover and the rover is driving away from it. Therefore, when the map is recentered, the rover may have already seen the region where the new intermediate goal is placed, and it is possible that there is an obstacle in this region. However, for this problem to occur, the rover must be attempting to navigate around a very large hazard and must drive large distances. Owing to rover power and drive time constraints, the distance that the rover can traverse in a single sol using AutoNav is limited. This limitation drastically reduces the chances of encountering this problem.

5.2. Coarse-Resolution Cost Maps

Another way to manage limited memory resources is to change the resolution of the cost map grid cells. Instead of constraining the cost map grid cells to be the same size as the goodness map grid cells, the cost map cells are allowed to be significantly larger. This allows fewer grid cells to cover the same area and thus results in smaller cost maps, which is what primarily dictates resource usage. Further, because the Field D* planner is able to compute paths that are not restricted to transitioning between grid cell centers or corners, it can be used to plan direct, low-cost paths even in very coarse-resolution grids.

Allowing for larger cost map grid cells does present some complications. Updating the cost map from the goodness map is now more difficult. Before, there was a one-to-one correspondence between cost and goodness grid cells, meaning that the goodness map could essentially be copied directly into the cost map. With larger cost cells, there are multiple goodness cells in each cost cell. In fact, there could even be fractional goodness cells in a given cost cell as shown in Figure 7. To simplify matters somewhat, the size of the cost cells is constrained to be an integer multiple of the goodness cells. This avoids splitting single goodness cells across multiple cost cells. Instead, each cost cell contains a fixed number of whole goodness cells. In this way, the complication of dealing with fractional cells can be avoided. However, a method is still needed to convert multiple goodness values into a single cost value.

One simple and safe method would be to use the minimum goodness value in a given cost cell to set the cost. In some cases this is not necessarily the best approach. Figure 8 illustrates what can happen when a narrow corridor is encountered. By using the minimum goodness value to update the cost value,

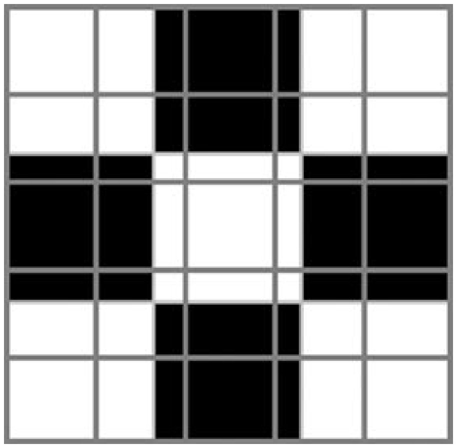


Figure 7. Goodness map overlaid on a cost map. Goodness grid cells are outlined in gray. Cost cells are shown in black and white. In addition to a complete goodness cell, each cost cell contains pieces of three, five, or even eight other goodness cells.

narrow corridors in the goodness map can become completely blocked in the cost map. One way to mitigate this problem is to employ a more lenient standard when updating cost cells containing obstacles. Cost cells that are less than half obstacle are set to the maximum traversable cost. Cost cells that are half obstacle or more are set to obstacle. This greatly reduces the chances of closing off narrow corridors.

Larger cost map grid cells also require a new strategy for handling unknown goodness cells. There is no traversability information in these cells, and previously they could just be ignored. The situation is more complicated when there are multiple goodness cells in each cost cell. One option for handling unknown goodness cells is to not update cost cells containing *any* unknown goodness cells. This is a less than ideal solution. A cost cell could contain many goodness cells with known values, but if there is one unknown value, all this information will be ignored. This situation happens frequently in cells at the edge of the cameras's field of view. To fully utilize the terrain assessment, the unknown goodness cells could merely be ignored and the minimum goodness of the populated goodness cells used to update the cost cell. This approach presents a more subtle problem, which is illustrated in Figure 9. During each step the rover takes, an area around the edge of the goodness map is set to unknown. This erases old data behind the rover to make room for new data in front of it. The problem arises when obstacle goodness cells behind the rover are set to unknown, but there are still some goodness cells in a given cost cell that have not been cleared and are not obstacle. The minimum goodness is therefore no longer obstacle, and if the minimum goodness is used to update the cost cell, the cost will be changed from obstacle to traversable. This causes Field D* to forget about obstacles, which is highly undesirable.

The solution to these problems is to make use of another value that is stored as part of the local terrain map. In addition to a goodness value, each goodness cell contains a certainty value as well. If the certainty is not zero, then the grid cell is in the current field of view and was just updated. It acts as a sort of new data flag. Therefore, if there is no certainty in a given cost cell, it is probably an old cell that is behind the

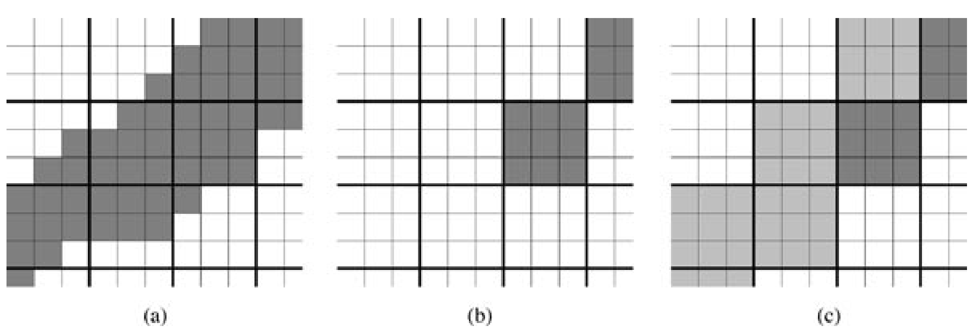


Figure 8. Here, each cost cell contains nine goodness cells (cost cells are outlined in heavy black). White represents obstacle, dark gray is traversable, and light gray is the maximum traversable cost. Coloring is by goodness value in (a) and cost value in (b) and (c). The cost map in (b) is produced by using the minimum goodness value in each cost cell. The cost map in (c) is produced using a more lenient update rule. If the cost cell is less than half obstacle, it is set to the maximum traversable cost instead of obstacle. Note that the corridor is blocked in (b) but not in (c).

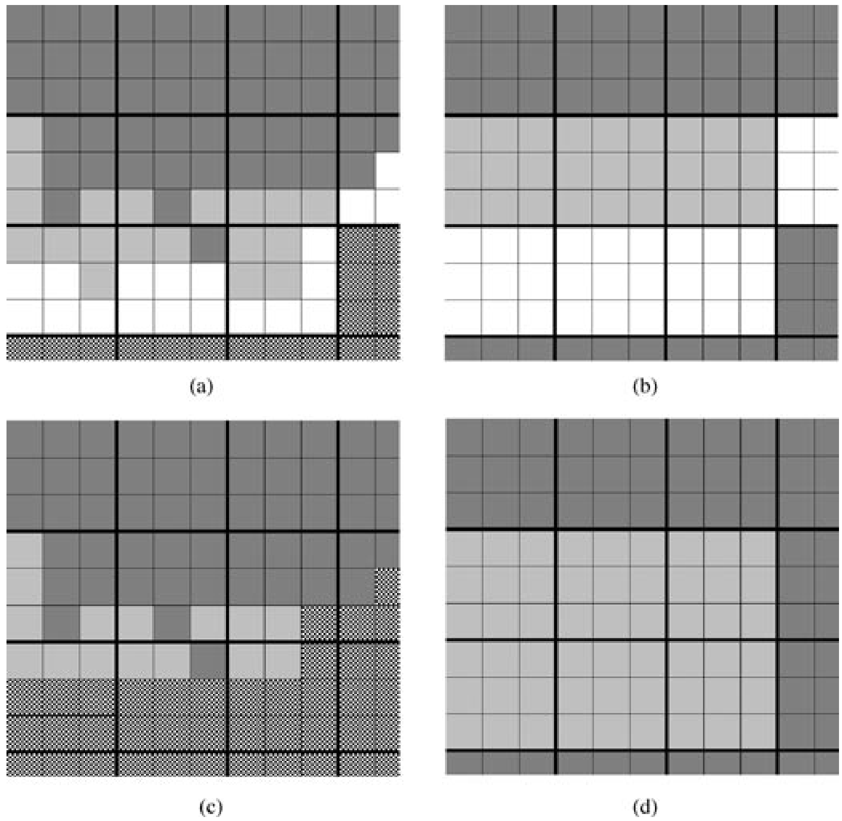


Figure 9. Here, each cost cell contains nine goodness cells (cost cells are outlined in heavy black). White represents obstacle, gray is traversable, and checkered cells are unknown. The goodness and cost maps for one step are shown in (a) and (b), respectively. Similarly, (c) and (d) are for the next step. Between steps, the rover has moved up to the left. The cost maps are produced using the minimum goodness value that is not unknown. Note that as the rover moves, obstacles are forgotten from the goodness map, and using this update strategy these obstacle cells are set to traversable in the cost map.

rover. In this case, the first approach is utilized, which is to not update the cell if it has any unknown goodness. This avoids forgetting data in the cost map. On the other hand, if there is certainty in a given cost cell, it contains new data and is probably in an area that has not been seen before. For these cells, the second approach is used, and the cost is updated using the minimum goodness value. In this way, new terrain assessments are added to the cost map as early as possible.

5.3. Map Filtering

In certain situations, the planning process for a given drive step can take more than an order of magnitude longer than usual. Recall that Field D* is used to plan a path from each arc endpoint to the goal. Also recall that Field D* will not plan paths through obstacle cells. If an arc endpoint happens to fall in an obstacle cell, the algorithm immediately returns, indicating that there is no path to the goal. However, planning time can swell if an arc endpoint falls in a nonobstacle region completely surrounded by obstacle cells, as shown in Figure 10(a). This is an artifact of how the planning process is carried out. The search begins at the goal location and expands outward. When the start state (the arc endpoint in our case) is reached, the search terminates and the path cost is returned. Unfortunately, when an arc endpoint falls into a small region completely surrounded by obstacles, it is in fact unreachable. For Field D* to make this determination, every reachable state in the cost map must be expanded. Instead of the usual relatively few states being expanded, all states in the cost map

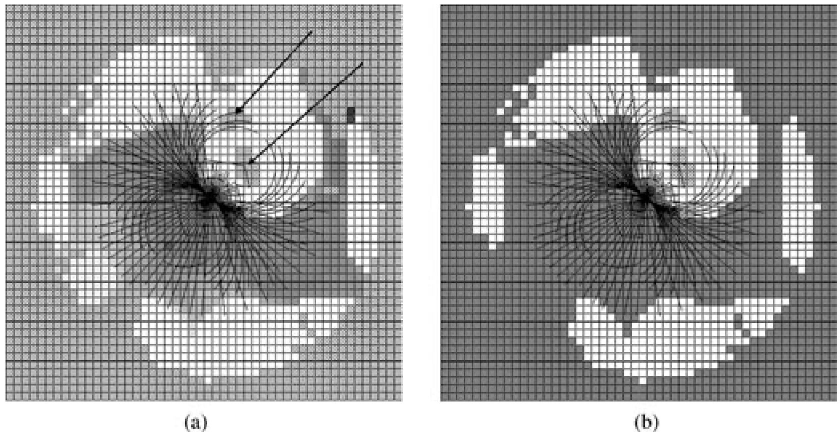


Figure 10. Goodness map filtering. Obstacle cells are shown in white. The goodness map in (a) contains regions completely surrounded by obstacle cells. Planning time is greatly increased when arc endpoints fall in these regions. All cells reachable from the rover location are shown in dark gray in (b). Note that the regions surrounded by obstacle are not marked as reachable.

are expanded, significantly increasing the planning time. This explosion in usage of already limited CPU resources is clearly undesirable.

To solve this problem, the goodness map is filtered before it is used to update the Field D* cost map. A flood-fill algorithm is used to identify all cells in the goodness map reachable from the current rover location. The rover location is first marked as reachable. Then each adjacent (eight-connected), nonobstacle cell is added to a list for later processing. Next, a cell is removed from the list and marked as reachable, and its new nonobstacle neighbors are added to the list. This repeats until the list is empty, indicating that all cells reachable from the rover location have been identified, as shown in Figure 10(b). Finally, all nonreachable, nonobstacle cells are set to obstacle. By doing this, arc endpoints that would have been problematic now end in obstacle cells. In this case, no planning is necessary to determine that no path to the goal exists.

Even though map filtering is done during every map update, in the long run it still saves time. As an added benefit, it also allows for much more consistent and predictable planning times. There are a couple of reasons why map filtering at every step is much faster than letting Field D* occasionally expand the entire cost map. First, map filtering is done on the goodness map, which is much smaller than the Field D* cost map. In addition, the simple flood-fill check for reachability takes much less time than the full Field D* planning process. In fact, the time necessary to filter the goodness map is negligible when compared even to the nominal planning time necessary for each drive step.

6. SSTB RESULTS

The MER surface system test bed (SSTB) was used to extensively test flight software modifications. The SSTB is a high-fidelity engineering model of the MERs. It is essentially identical in form and electromechanical function to Spirit and Opportunity, with a few minor exceptions. The SSTB has no solar panels, and some of its electronics are housed in an adjacent clean room. A physical tether provides a link between the rover and these electronics. The tether also provides power to the rover. The SSTB is housed in an indoor sandbox approximately 9 m wide and 22 m long (Litwin, 2005). A ramp tilted at 25 deg occupies one end. The SSTB and its test environment is shown in Figure 11.

GESTALT alone performs well in simple situations, including navigation in areas free from hazards and navigating around small discrete obstacles. However, the real strength of Field D* is navigation in much more complex situations. Unfortunately, the relatively small size of the sandbox makes constructing complex obstacle arrangements difficult. Testing was limited to this environment for several reasons. The SSTB was the only available system with

Figure 11. MER SSTB rover in an indoor sandbox at JPL.

enough fidelity to perform flight software testing requiring imaging and driving, with the driving decisions based on imaging results. It was infeasible to move the rover to a larger outdoor environment due to the tremendous effort that would be necessary to move all the support equipment needed to run the rover (remember that most of the rover electronics are actually housed in a clean room adjacent to the sandbox). However, even in the limited sandbox environment, constructing situations for which GESTALT alone fails to reach the goal is not difficult. For instance, navigating around a cul-de-sac obstacle arrangement is nearly impossible for GESTALT alone. Figure 12 illustrates a situation with not one, but two, cul-de-sacs. Owing to the limited size of the sandbox, the goal is placed outside the sandbox and is not actually reachable. Figure 12(a) shows the initial position of the rover. The rover begins by driving straight into the first cul-de-sac. The rover reaches the bottom of the cul-de-sac in Figure 12(b). Up to this point, the behavior with and without Field D* was roughly equivalent. However, with GESTALT alone the rover became stuck here. Field D*, on the other hand, plans a path around the first cul-de-sac and into the second. The rover is then guided into the second cul-de-sac as shown in Figure 12(c). Once the determination is made that there is no route through the second cul-de-sac, the rover drives back toward the only unexplored region of the sandbox as shown in Figure 12(d). Eventually Field D* fails, indicating that no paths to the goal exist.

Over the course of testing, Field D* was used to guide the SSTB toward roughly 100 different goal locations. Initially, a variety of simple tests were completed. In an obstacle-free setting, the rover was placed at a variety of different initial headings relative to the straight line to the goal. Navigation through a field of traversable rocks was tested. Navigation around a single rock in various positions relative to the path between the rover and the goal and navigation between two rocks separated by a variety of distances were also tested. More complex obstacle arrangements in which GESTALT alone would almost certainly fail to guide the rover to the goal were tested as well. Situations were constructed necessitating navigation into and out of single or multiple cul-de-sacs. In addition, lines of rocks were used to produce an arrangement similar to the one shown in Figure 2. Overall, the performance was extremely good. In the vast majority of cases the rover was able to reach the goal when Field D* was used, and performance in the simple test cases was at least as good as with GESTALT alone. Surprisingly, one of the biggest problems faced during testing was goal placement. If the goal is placed in an obstacle cell, Field D* is unable to plan any paths. When the rover gets close enough to the goal to determine that it is in an obstacle cell, Field D* will fail. Although the rover does not reach the goal, this should not necessarily be considered an AutoNav failure. In these situations the goal location is not safe, and the rover should not drive onto it. Owing to the very limited space in the sandbox, squeezing the goal location into a safe area (after all obstacles have been expanded by the rover radius) was sometimes a challenging proposition.

With Field D*, the rover is able to explore the environment much more fully when attempting to locate a path to the goal. This allows the rover to almost always arrive at reachable goals. The downside, of course, is the increased resource utilization required. For Field D*, the additional CPU time and memory usage are fairly minimal. Much of the testing was done using 50×50 m cost maps. The cost cells were 40×40 cm, which is twice the resolution of the goodness cells. Almost no difference was noticed in rover behavior when moving from 20- to 40-cm cost cells. With these settings, Field D* utilizes less than 1 Mbyte of memory. In addition, each drive step takes only about 3% longer when Field D* is enabled. Even with these very modest requirements, Field D* is able to significantly improve onboard AutoNav capability.

7. MARS SURFACE TESTING

Before being approved for general use on Mars, Field D* underwent five carefully planned checkouts. The

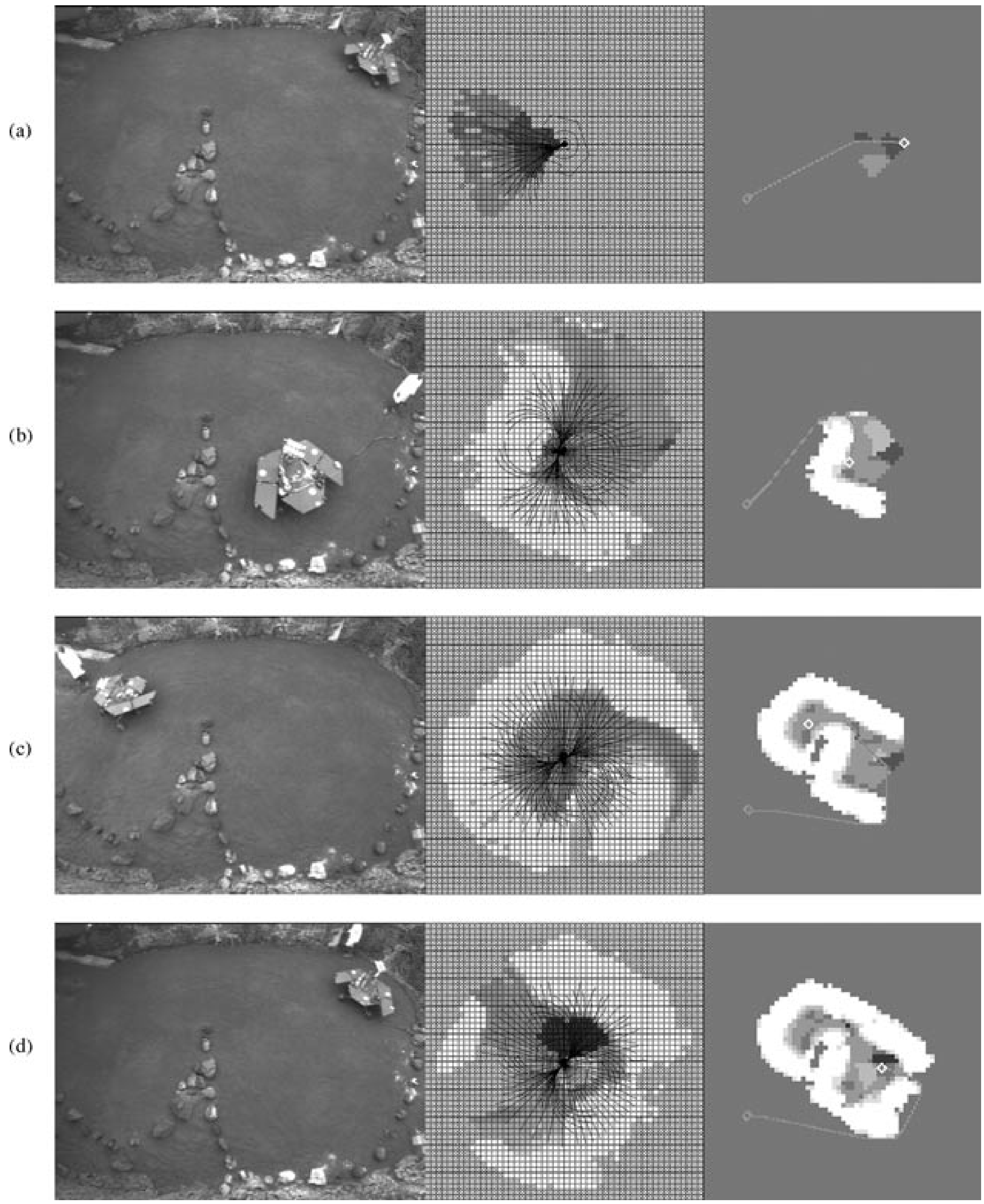


Figure 12. Field D*-assisted hazard avoidance using the SSTB. The left-hand image is an overhead view of the sandbox. The middle image is the local goodness map, and the image on the right is the Field D* cost map. Note that the entire goodness map is shown, but only a portion of the cost map is included. Checkered cells have unknown traversability. All other cells are colored based on a gradient between black (high goodness/low cost) and white (low goodness/high cost). The checkered line on the cost map is the path planned between the rover and the goal. The size of each goodness cell is 20×20 cm. Each cost cell is 40×40 cm.

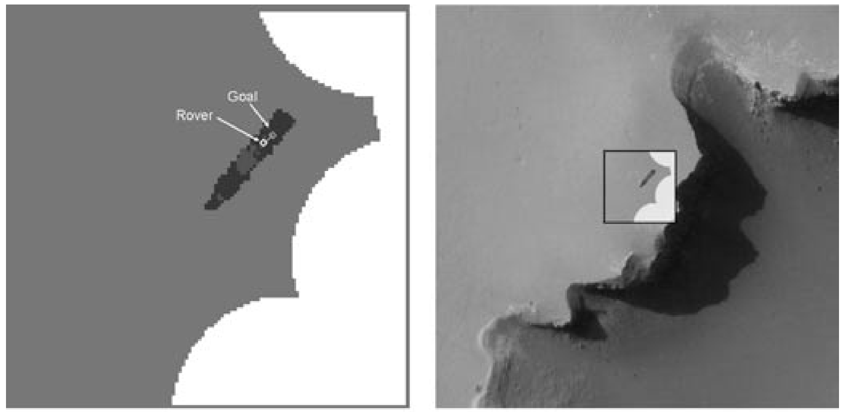
checkouts were designed to incrementally verify that the software works as expected, beginning with very basic, low-risk tests and moving toward a complete confirmation of all features. These tests were designed not only to verify that Field D* operates correctly but also to ensure that other software modules function as expected when Field D* is enabled. All tests were performed on Opportunity, due to the nonfunctional drive motor on Spirit's right front wheel. This broken actuator results in the wheel being locked in the brake position, which makes driving much more challenging. The position of a rover after a commanded arc is highly dependent on terrain interaction. With one locked wheel, Spirit generally experiences a significant level of slip. This makes position estimates based on wheel odometry unreliable, and therefore visual odometry is almost always used to obtain more accurate position estimates (Cheng et al., 2005). Unfortunately, due to the slow speed of the processor onboard the rovers, the time needed to compute visual odometry updates significantly limits how far Spirit can drive in a single sol. AutoNav is generally used when traversing areas not yet seen by engineers on Earth. It is a very rare occasion that Spirit drives far enough in a sol to necessitate activating AutoNav. This, combined with the fact that commanded arcs (including those selected by AutoNav) are rarely executed as expected (due to the right front wheel being dragged), makes Spirit a poor choice for Field D* testing.

Although most of the testing on SSTB was done using the belly-mounted hazard avoidance cameras, Opportunity generally uses the navigation cameras on the mast when driving autonomously. The region of Mars where Opportunity is located is generally very sandy with few rocks. The hazard avoidance cameras are unable to pick up the fine texture of the sand, and therefore the stereo correlator has few features to match upon. As a result, stereo range maps are often extremely sparse. The navigation cameras, on the other hand, are higher resolution and able to pick up the sand's fine texture. The downside is that the field of view is much narrower for the navigation cameras. Whereas the hazard avoidance cameras can see almost 180 deg, the navigation cameras are limited to about 45 deg. However, it is possible to capture multiple navigation camera images with different camera pointings to fill in a larger field of view. Of course stereo processing must be run on each image pair, which can significantly slow the rate at which the rover can drive. To help speed things up, the rover does not have to update its map after every single drive step. The rover can drive for a limited number of steps without imaging, provided that it has not located any obstacles nearby. When doing this, the rover still generates votes at every step to select the next arc but does not update its world map. Instead, it must stay within the area deemed safe in the most recently available map. This technique can significantly increase the speed at which the rover is able to drive autonomously. For each checkout, a 50×50 m cost map was utilized, with a grid cell resolution of 40 cm. The local goodness map was 12×12 m with a grid cell resolution of 20 cm.

7.1. Checkout One: Backseat Driver

On sol 1014, Field D* was run for the first time on Mars. For this initial test, Field D* was placed in a special "backseat driver" mode. In this mode, Field D* builds a cost map, does its planning, and creates its votes. However, these votes are not actually used to influence the arcs selected and executed by the rover. Instead, the standard AutoNav system is in complete control and does not use any information generated by Field D*. The Field D* telemetry (including cost maps and votes) is sent back to Earth, where engineers can verify that Field D* was operating nominally and generating appropriate steering recommendations.

The first checkout was conducted in a very flat area completely free from hazards. This was done to minimize risk to the rover should something go wrong with the test. A goal was chosen 12 m straight ahead of the rover with a tolerance of 2 m, for an expected drive of 10 m. Only a single heading for the navigation camera was used when updating the map, and the rover was allowed to drive up to four steps of 50 cm each per map update. Generally, a single navigation camera heading provides too limited a field of view and is insufficient for driving through unknown terrain. However, in this case, the rover was driving through terrain known to be completely safe, and therefore a fixed camera pointing for each update was sufficient. Figure 13 shows the final Field D* cost map. The map was updated six times during the traverse, and a total of 21 drive steps were taken before successfully reaching the waypoint. The dark areas represent the region imaged and filled in during the traverse. The white regions are keep-out zones set by engineers on Earth. Keep-out zones are areas that should be avoided by the rover. Should the rover ever accidentally stray into one, the mobility software will prevent any further driving. These keep-out zones were placed to prevent the rover from getting too close to the edge of Victoria Crater should something go wrong. Although Field D* was not in control of the rover during the test, all other subsystems functioned nominally with the Field D* software running. All Field D* telemetry was generated



(a) (b)

Figure 13. Cost map from the first Field D* checkout. The Field D* cost map from the final step is shown in (a). The rover is depicted as a white diamond and the goal as a checkered diamond. The dark areas were imaged and determined to be safe by the rover. The white regions are keep-out zones used to keep the rover away from Victoria Crater. An orbital image captured by the Mars Reconnaissance Orbiter is shown in (b), along with an overlay of the Field D* cost map. Courtesy NASA/JPL-Caltech.

as expected. During the majority of steps, the Field D* votes were highest for the straight-ahead arc. In every other case, the strongest preference was for the shallowest right or left turn. This is exactly as expected, and the checkout was deemed a success.

7.2. Checkout Two: Phantom Hazard

The second Field D* checkout was run on sol 1082. This was the first time Field D* was actually used to influence the drive path taken by the rover. The test was set up to verify hazard avoidance capability without placing the rover in danger should something go wrong. The test was done in an area completely free from of hazards; however, a 2-m-wide keep-out zone was added approximately 5 m straight ahead of the rover. The AutoNav system treats keep-out zones like any other hazard and attempts to drive around them. A waypoint was selected 9 m straight ahead of the rover on the opposite side of the keep-out zone. Generally, using only a single navigation camera pointing direction to update the map is insufficient when driving through hazardous regions. To save time, and because the area was completely free from any real obstacles, a fixed pointing for the navigation camera was used for each update. As in the first test, the rover was allowed to drive up to four 50-cm steps between map updates.

The first half of the drive went well. The rover immediately selected a path to the right and turned slightly throughout the first half of the drive in order to safely get around the keep-out zone. However, during the second half of the drive, the rover did not turn back toward the goal as sharply as expected. Figure 14 shows an overhead view of the drive. Notice that during the second half of the drive, the rover did not take a course directly back toward the waypoint. However, it did turn back enough that it was able to successfully come within the specified 2-m tolerance around the goal. There are a couple of reasons for this behavior. The first was the limited field of view caused by using only a single image to update the navigation map. The AutoNav system will not allow the rover to drive into regions that it has not seen and verified to be safe. Turning sharply can quickly move the rover out of the narrow region verified to be safe. This is not allowed, and thus the arc set available for execution was significantly restricted. The second factor impacting the shallow turning during the latter half of the drive was the incorrect setting of one of the Field D* parameters. This parameter was thought to have been changed to the correct value on a previous sol when many of the other navigation parameters were set. It was discovered late in the planning process (too late to correctly set the parameter) that this was not the case. Because there was no risk to vehicle safety from this parameter setting, it was decided to go ahead with the checkout instead of pulling the whole thing. This parameter setting led to suboptimal sampling of the

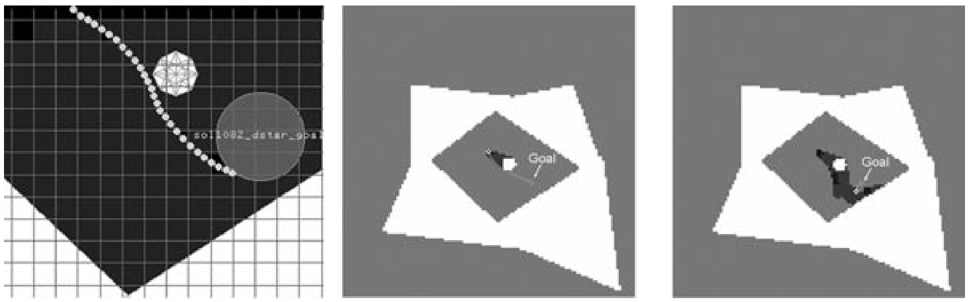


Figure 14. Maps from the second Field D* checkout. An overhead view of the drive is shown in (a). The white regions are keep-out zones, and the light gray dots show the path taken. The dark gray circle represents the goal location and its 2-m tolerance. The initial and final Field D* cost maps are shown in (b) and (c), respectively. The checkered line shows the path planned by Field D*, and the white regions are keep-out zones. Note the keep-out zones completely surrounding the test area to keep the rover from straying too far off course if something were to go wrong.

arc endpoints and contributed to the shallow turns at the end of the drive. Part of the checkout process is to uncover and correct these kinds of issues.

7.3. Checkout Three: Everything Together

The third checkout was the most difficult for a variety of reasons. The purpose of this checkout was to ensure that Field D* could be used to avoid obstacles detected by GESTALT. It was difficult to locate a suitable area for this test. All of the checkouts were performed as Opportunity made its way around the rim of Victoria Crater. Unfortunately for this checkout, there are very few rocks in this area. It is mostly an extremely flat, sandy plain. This test required a rock short enough to be safely driven over but tall enough to be detected as an obstacle. By using such a rock, if something were to go wrong with the test and the rover did not avoid the hazard, no damage would be done. Fortunately, Opportunity happened upon a small crater surrounded by a variety of rocks (see Figure 15). Usually the obstacle detector classifies objects taller than 20 cm as obstacles. For this test, that threshold was lowered such that the rocks to the right of the crater were classified as obstacles.

This checkout was conducted on sol 1160. A goal was chosen 11 m straight ahead of the rover, on the other side of the crater. Keep-out zones were added to cover the crater, as well as some of the larger rocks. The rover would have been able to detect and avoid these obstacles on its own, but because this was a checkout, keep-out zones were added for an extra level of safety. Because the rover needed to drive around detected obstacles, two navigation camera pointings were used to update the maps at each step. Imaging to update the navigation map was done before each and every 50-cm drive step. In addition, visual odometry was used to more accurately track the rover's position during the drive. This was done to facilitate precise pointing of the cameras at the end of the drive in order to capture imagery of a science target. Combining keep-out zones, stereo vision, hazard detection, navigation with Field D*, and visual odometry represents the largest simultaneous use of autonomy on either rover to date.

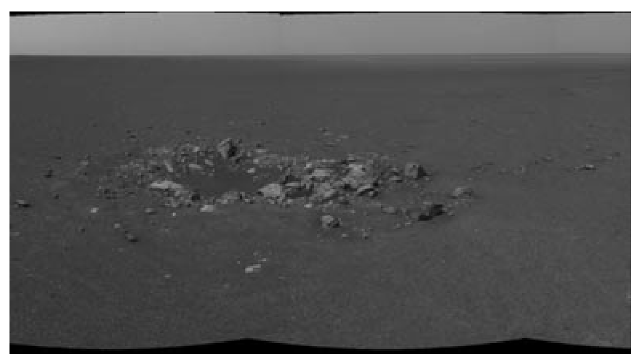


Figure 15. Location of the third checkout. Hazard detection parameters were modified to make the small rocks to the right of the crater appear to be obstacles.

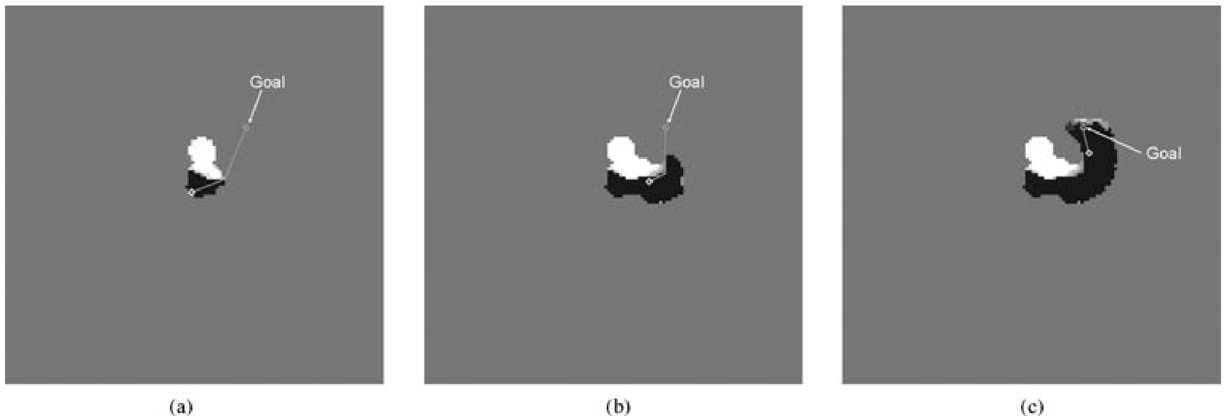


Figure 16. Cost maps from the third Field D* checkout. The initial and final cost maps are shown in (a) and (c), respectively. A map from the middle of the drive is shown in (b). The checkered line represents the path planned by Field D*, and the white regions are hazards. The dark areas are safe.

Cost maps from this checkout are shown in Figure 16. From the start, the area several meters ahead of the rover is identified as hazardous. For the first five drive steps, the rover selected a series of moderately sharp right turns to avoid these obstacles. For the next eight steps, the rover drove almost straight, using a sequence of very slight left turns to follow the contour of the detected hazards. Next, a series of seven slightly sharper left turns were used to turn back toward the goal location as the rover makes its way around the end of the rock field. With the goal almost straight ahead, the rover selected nearly straight arcs for the final segment of the drive. The rover was commanded to drive to within 2 m of the goal location. However, the allotted time for the drive ran out with the rover 3.3 m from the goal (all that autonomy takes a lot of time!). Before driving away from the test site on sol 1162, a panorama was taken looking back on the drive path (Figure 17). Overall the checkout went flawlessly, with the rover smoothly and efficiently making its way around the hazards and toward the goal. In addition, without Field D* there is a significant possibility that the rover would have become stuck at the leading edge of the rock field, unable to find its way around.

7.4. Checkout Four: Going the Distance

With the ability to safely navigate around hazards verified, it was time to test a long-distance drive. This

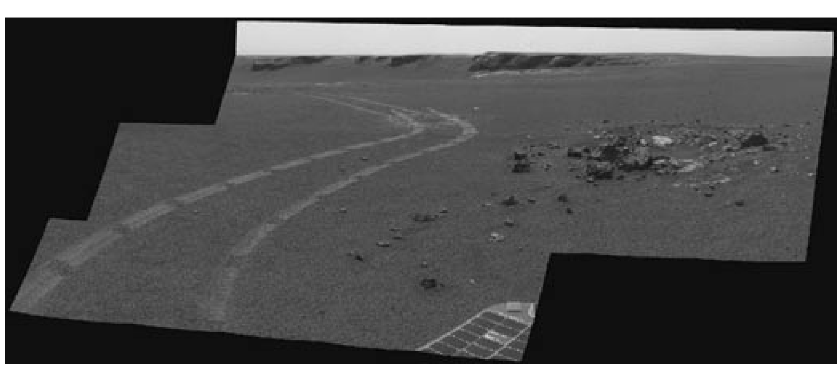


Figure 17. Mosaic looking backward on the tracks of the third Field D* checkout. Courtesy NASA/JPL-Caltech.

checkout had several purposes. The first was to verify that there was no degradation in system performance over many steps. This checkout would also verify that a goal location outside the bounds of the Field D* cost map could be chosen without problem. Finally, the ability to recenter the cost map as the rover reached the edge of the map would also be tested. On sol 1188, a 40-m drive was planned, which was broken up into two 20-m segments. In between the two segments, a wheel slip test was performed. The immediate area around the rover was flat and free from hazards, but due to the length of the drive, it was difficult to verify that there were no obstacles near the goal location. Therefore, it was assumed that the rover might encounter obstacles that it would need to autonomously avoid. That being the case, two navigation camera pointings were used for each map update, and the rover was allowed to drive only three 50-cm steps between map updates.

The drive began with the rover facing the goal location. During the first 20-m drive segment, Field D* consistently preferred straight arcs or very slight turns. However, the drive path actually taken by the rover was not exactly straight. On several different occasions the hardest right or left turn was taken. This was due to the sharp turn correction feature built into the vote arbitrator. If the sharpest turn is included in the set of highly ranked arcs, then it is selected. Generally, this feature helps the rover aggressively turn to get around obstacles, but it also sometimes leads to the weaving behavior seen here. Figure 18 shows the view from the rear hazard avoidance camera after the first leg of the drive. Note the snaking drive path in the distance.

At this point in the drive, the slip check was conducted. During a slip check, the rover is commanded to drive a short distance, and visual odometry is used to measure the actual distance that was traveled. If the rover did not make enough progress, then no further driving is allowed. Slip checks are done at regular intervals in order to ensure that the rover is not getting bogged down and digging itself further and further into a sand trap. Visual odometry works by tracking image features between frames to estimate rover motion. Unfortunately, the flat, mostly featureless plains around Victoria Crater make finding features difficult. Here, the rover's own tracks are usually the only workable features. Therefore, the navigation cameras are usually pointed toward the tracks for the slip check. Alas, the last drive step before the slip check happened to be the sharpest left turn. This

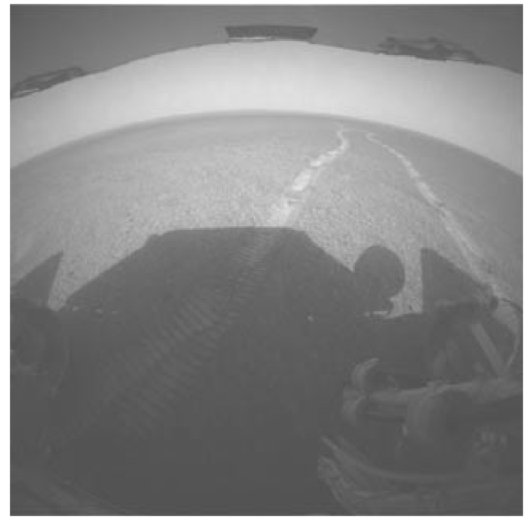


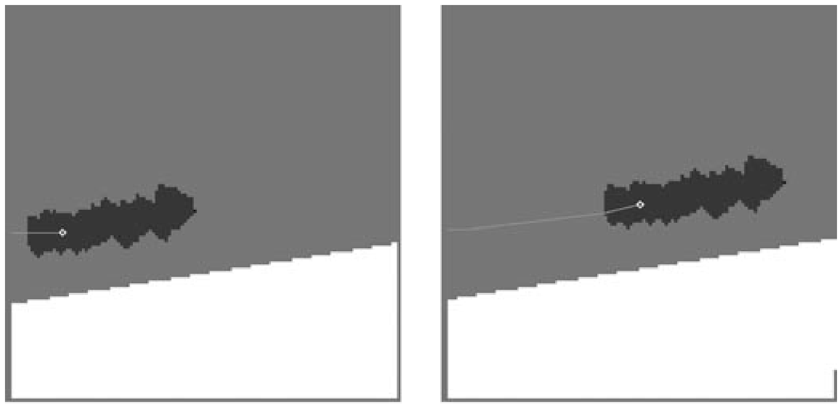
Figure 18. Image capture by the right rear hazard avoidance camera following the fourth checkout.

meant that the tracks were no longer directly behind the rover as expected and were not in the field of view when visual odometry was run. Not finding any features, visual odometry failed to produce a motion estimate during the slip check. Because it could not be determined whether the rover was bogging down, no further driving was allowed.

Even though only half of the planned drive was executed, the test was deemed a success. Fortunately, right before the last step, the Field D* map was recentered. This meant that all objectives of the test had been completed. Figure 19 shows the cost map before and after the map was recentered.

7.5. Checkout Five: Preconceived Notions

Usually, the rover begins with a blank map, which is updated as the rover sees new areas. However, it is also possible to create an initial map on Earth and upload that to the rover. This can give the rover a better idea of what is coming up beyond its detection range or field of view. The map is still updated autonomously based on what the rover senses; however, a more accurate initial map can help the rover plan better paths. The purpose of the fifth and final checkout was to verify this map upload ability. This test was done on sol 1200, following a 60-m blind drive. Because the area in which the checkout was to take place was so far away, it was difficult to say for certain that the area was hazard free ahead of time.



(a) (b)

Figure 19. Cost maps from the fourth Field D* checkout. As the rover approaches the edge of the map (a), it is autonomously recentered on the rover position (b). The planned path is shown as a checkered line (the goal was selected outside the map bounds). The white region is a keep-out zone placed between the rover and Victoria Crater. The dark areas were verified safe by the rover during its traverse.

Therefore the standard setup for driving through possibly hazardous terrain was used (two navigation camera pointing directions for each map update and no more than three 50-cm steps between updates). Figure 20 shows the map that was created and uploaded to the rover. Victoria Crater was off the right side of the map, and therefore an obstacle region was added along that side of the map. A goal was chosen straight ahead of the rover, off the bottom of the map. Obstacles were placed in the map between the rover and the goal. This was done with the expectation that the rover would make right-hand turns in order to get around the end of this obstacle region. This would serve as verification that the map was in fact being utilized in the path selection process.

This drive was limited to 10 50-cm steps. The drive began with the rover taking a sharp right-hand turn. For the rest of the drive, Field D* preferred shallow to moderate right-hand turns in order to line the rover up to navigate around the end of the obstacle region. This test was deemed a success. The downlinked map data products clearly showed that the map had been loaded as expected. The fact that the rover turned right during the drive demonstrated that the rover was influenced by the uploaded map and the obstacle regions placed far beyond its detection range.

With all five checkouts completed successfully, Field D* was approved for everyday usage. Shortly thereafter, Opportunity entered Victoria Crater. The high slopes along the ingress path necessitate very cautious driving. Therefore the drives are generally fairly short, and it would be imprudent to drive beyond the areas that had been fully seen and analyzed by engineers on Earth. This makes AutoNav with hazard avoidance unnecessary, and Field D* has yet

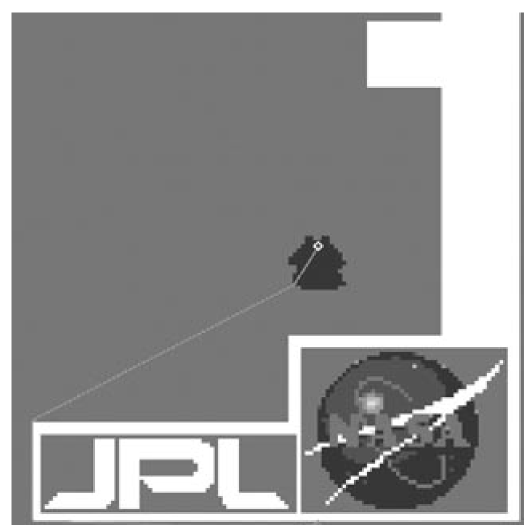


Figure 20. Field D* cost map uploaded to the rover for the fifth and final checkout. The initial rover position is represented as a white diamond. The path planned by Field D* is shown as a checkered line. White areas are obstacles, and dark regions are safe.

to be used on Mars (as of sol 1520) beyond the checkouts. However, once the Victoria science campaign is over and the rover leaves the crater, it is likely that Field D* will prove useful for long drives across the plains.

8. CONCLUSIONS

Autonomous hazard avoidance by the MERs using the GESTALT local planner keeps the rovers safe and works well in the presence of simple discrete obstacles. However, it is susceptible to failure when more complex hazard arrangements are encountered. To address this shortcoming, the hazard avoidance system was augmented with a global path planner. Field D* was integrated into the MER flight software and uploaded to Spirit and Opportunity during the summer of 2006 as part of a significant software upgrade. Field D*-assisted hazard avoidance was extensively tested using the SSTB before the upload and has been validated on the surface of Mars. Obstacle avoidance is at least as good as with GESTALT alone and in many cases much better. Field D* allows the rover to much more robustly navigate around hazards. With Field D*, the rover is less prone to getting stuck and can reach long-range goals even when faced with complex hazards.

ACKNOWLEDGMENTS

The research described in this paper was performed at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under a contract with the National Aeronautics and Space Administration (contract NAS7-03001) and at Carnegie Mellon University (CMU) under a contract with the JPL (contract 1263676). This research was funded by the MER project and the Mars Technology Program (MTP) under task order NM0710764. CMU configured Field D* for MER and provided it to JPL under a research license with the California Institute of Technology, and JPL integrated Field D* into the MER flight software. CMU's work was performed under the MTP Reliable and Efficient Long-Range Autonomous Rover Navigation task, and JPL's work was performed under the MTP D* Integration into MER task. The authors would like to thank Mark Maimone for his sage advice during the integration and testing process. Without his assistance this process would have been infinitely more difficult.

REFERENCES

Biesiadecki, J., & Maimone, M. (2006, March). The Mars Exploration Rover surface mobility flight software: Driving ambition. In 2006 IEEE Aerospace Conference Proceedings, Big Sky, MT.

Biesiadecki, J. J., Baumgartner, E. T., Bonitz, R. G., Cooper, B. K., Hartman, F. R., Leger, P. C., Maimone, M. W., Maxwell, S. A., Trebi-Ollenu, A., Tunstel, E. W., & Wright, J. R. (2005, October). Mars Exploration Rover surface operations: Driving Opportunity at Meridiani Planum. In 2005 IEEE International Conference on Systems, Man, and Cybernetics, Waikoloa, HI (pp. 1823–1830).

Brock, O., & Khatib, O. (1999, May). High-speed navigation using the global dynamic window approach. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Detroit, MI (pp. 341–346).

Cheng, Y., Maimone, M., & Matthies, L. (2005, October). Visual odometry on the Mars Exploration Rovers. In 2005 IEEE International Conference on Systems, Man, and Cybernetics, Waikoloa, HI (pp. 903–910).

Ferguson, D., & Stentz, A. (2006). Using interpolation to improve path planning: The Field D* algorithm. Journal of Field Robotics, 23(2), 79–101.

Goldberg, S., Maimone, M., & Matthies, L. (2002, March). Stereo vision and rover navigation software for planetary exploration. In 2002 IEEE Aerospace Conference Proceedings, Big Sky, MT (pp. 2025–2036).

Greco, M., & Snyder, J. (2005, October). Operational modification of the Mars Exploration Rovers' flight software. In 2005 IEEE International Conference on Systems, Man, and Cybernetics, Waikoloa, HI (pp. 8–13).

Kelly, A. (1995). An intelligent predictive control approach to the high speed cross country autonomous navigation problem. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Kelly, A., Stentz, A., Amidi, O., Bode, M., Bradley, D., Diaz-Calderon, A., Happold, M., Herman, H., Mandelbaum, R., Pilarski, T., Rander, P., Thayer, S., Vallidis, N., & Warner, R. (2006). Toward reliable off road autonomous vehicles operating in challenging environments. International Journal of Robotics Research, 25(5–6), 449–483.

Kolski, S., Ferguson, D., Bellino, M., & Siegwart, R. (2006, June). Autonomous driving in structured and unstructured environments. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Tokyo, Japan (pp. 558–563).

Leger, P. C., Trebi-Ollennu, A., Wright, J. R., Maxwell, S. A., Bonitz, R. G., Biesiadecki, J. J., Hartman, F. R., Cooper, B. K., Baumgartner, E. T., & Maimone, M. W. (2005, October). Mars Exploration Rover surface operations: Driving Spirit at Gusev Crater. In IEEE International Conference on Systems, Man, and Cybernetics, Waikoloa, HI (pp. 1815–1822).

Litwin, T. (2005, October). General 3D acquisition and tracking of dot targets on a Mars rover prototype. In

2005 IEEE International Conference on Systems, Man, and Cybernetics, Waikoloa, HI (pp. 443–449).
Reeves, G., & Snyder, J. (2005, October). An overview of the Mars Exploration Rovers' flight software. In 2005 IEEE International Conference on Systems, Man, and Cybernetics, Waikoloa, HI (pp. 1–7).
Schenker, P. (2006, March). Advances in rover technology for space exploration. In 2006 IEEE Aerospace Conference Proceedings, Big Sky, MT.
Singh, S., Simmons, R., Smith, T., Stentz, A., Verma, V., Yahja, A., & Schwehr, K. (2000, April). Recent progress in local and global traversability for planetary rovers. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA (pp. 1194–1200).
Stentz, A., & Hebert, M. (1995). A complete navigation system for goal acquisition in unknown environments. Autonomous Robots, 2(2), 127–145.