2008

Continuous-Field Path Planning with Constrained Path-Dependent State Variables

G. Ayorkor Mills-Tettey
Carnegie Mellon University

Anthony Stentz
Carnegie Mellon University

M. Bernardine Dias
Carnegie Mellon University

Continuous-field path planning with constrained path-dependent state variables

G. Ayorkor Mills-Tettey, Anthony Stentz and M. Bernardine Dias

***Abstract*— Planning autonomous long range traverses for planetary exploration and similar robotic missions requires several important capabilities. These include the ability to generate smooth, optimal paths, the ability to reason about constrained path-dependent state variables such as energy, and the ability to replan rapidly in response to new information. Existing path planning approaches provide one or two of these capabilities but fall far short of supporting all three. We present a *framed cells* approach to path planning which enables the computation of smooth paths in state spaces that include constrained path-dependent variables. The effectiveness of this approach is demonstrated in simulation and on two different robots.**

I. INTRODUCTION

Future planetary exploration missions, such as the Mars Science Laboratory [1], call for a highly capable class of rover to explore and collect scientific data at distinct sites separated by distances of several kilometers. These ambitious exploration campaigns will be enabled by technology advances in many areas, including the development of algorithms for efficient and reliable autonomous long-range navigation. Autonomous kilometer-scale navigation requires path planning algorithms that can generate optimal or very high quality paths with respect to a metric of interest. While several existing algorithms produce paths that are optimal in a discretized planning space, such as a regular grid, there is a need to extend these to the continuous realm in order to produce results that are closer to the true optimal path, a feature described as *continuous-field path planning* [2]. In addition, planning for longer distances and at a higher level of abstraction requires reasoning about global, mission-relevant constraints involving parameters such as time, energy and risk. Furthermore, to be useful in unknown or partially known environments, these algorithms must be capable of responding quickly to unforeseen changes in the environment or the mission. In short, three important capabilities required of these path planning algorithms are (1) support for *continuous-field path planning*, (2) the ability to reason about constrained *path-dependent state variables* such as time, energy and risk, and (3) support for *rapid replanning* in response to changes in the problem. We elucidate these capabilities below.

This work was supported by the Jet Propulsion Laboratory, under contract "Reliable and Efficient Long-Range Autonomous Rover Navigation" (contract number 1263676, task order number NM0710764).

The first author is a Ph.D. student and the next two authors faculty at the Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA, (email: {gertrude, axs, mbdias}@ri.cmu.edu)

A. Continuous-field path planning

A common practice when planning paths in large unstructured environments is to represent the environment by a cost map specifying terrain traversal costs in different areas. Path planning for a mobile robot can be achieved by searching for a path through the space of feasible configurations or states of the robot, where state transition costs are computed using information from the cost map. The set of possible positions of the robot is often restricted to a finite size by dividing the world into a regular grid and considering only positions at the centers or corners of the grid cells, with each valid position having eight neighboring positions. This so-called eight-connected planning restricts path headings to multiples of $\pi/4$ and results in paths that are optimal in terms of the grid, but suboptimal with respect to the underlying environment. To obtain a result that is closer to the true optimal path, the algorithm must allow a larger range of path headings, thus relaxing the restriction that paths must pass through the corners or centers of grid cells. This is illustrated in Fig. 1 where the optimal eight-connected path between the start and the goal, shown by a solid line, deviates significantly from the true optimal path, shown by a dashed line.

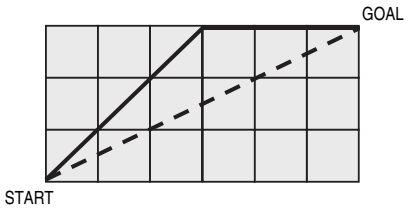

Fig. 1. Comparison between the optimal eight connected path (solid) and the true optimal path (dashed) in a uniform cost field.

B. Path-dependent state variables

At the basic level, path planning involves reasoning about the position of the robot. In some domains, however, an essential part of the state description, in addition to the robot's position, is one or more parameters whose value is a function of the path taken to reach that state. For example, in the case of a rover that uses solar power to charge its batteries, the currently available battery energy may be a necessary part of the robot's configuration or state description because the available energy determines what actions are feasible for the rover. Similarly, for an exploration robot that uses sensors to collect data from its environment, the

amount of space available in its data buffer may be included in the specification of its configuration, since this determines whether data collection actions are feasible at a given time and position. Parameters such as the robot's available energy or available storage space differ from parameters such as position in that the values of the former are a function of the path taken to reach the given state and are computed along with the objective function during planning, whereas the values of the latter are independent of the specific path taken. For example, given a start state A and a goal state B, the position of B is independent of the path taken from A to B whereas the energy available at B will differ depending on whether B is reached via a straight line from A or via a circuitous route. We call parameters such as battery level or disk capacity "dependent parameters" or "path-dependent state variables".

In a given application, there might be constraints on the path-dependent state variables. For example, when planning a path for a rover that uses solar energy to charge its batteries, the energy available to the rover is constrained by battery capacity and available solar flux. Similarly, for a planetary exploration rover that collects data from its environment and transmits it to a satellite or to earth as communication opportunities allow, available data storage space is constrained by disk capacity and opportunities for communication. In both of these examples, time is also a path-dependent state variable because the time at which the robot reaches a given position depends on the path taken to reach that position. In addition, energy costs for navigation or science activities as well as energy gains due to solar charging are position and time dependent, as are opportunities for communication to offload data.

Thus, planning involves searching for a path in a multi-dimensional state space; some dimensions of space correspond to independent parameters of the robot's configuration (such as x and y), while other dimensions correspond to dependent parameters (such as time and energy).

C. Rapid re-planning

A mobile robot traversing an unstructured environment may discover, via its sensors, discrepancies between its environment and the cost map used for planning. This is particularly true if little is known a priori about the environment, as may be the case in planetary exploration. Further, a robot following a spatio-temporal plan may fall behind schedule or find that it is ahead of schedule. In any of these situations, a new plan will have to be computed for the remaining unachieved goals in the mission. Given that this needs to be done in real time, it is advantageous to be able to re-compute the new optimal plan rapidly, using information from the previous planning session, rather than to compute the new optimal plan from scratch.

Several existing path planning algorithms, such as D* [3], Field D* [2] and TEMPEST/ISE [4], which all make use of cost maps and are suitable for path planning in unstructured environments, each provide one or more of these capabilities, but not all three. In this paper, we present a novel approach to path planning that encompasses these three important capabilities. The next section presents related work and highlights the strengths and shortcomings of existing algorithms. We then present details of the problem of interest, our approach, and experimental results from simulations and robotic field tests. Finally, we present our conclusions and future work.

II. BACKGROUND AND RELATED WORK

The planning paradigm used in this work is that of a backwards search from the goal to the start. The goal is assigned an objective function value of zero, and the objective function value of any other node s represents the cost of the optimal path from s to the goal. In general, the objective function value of a node is computed by minimizing over its neighboring nodes, the sum of the objective function value of the neighboring node and the transition cost to neighboring node. Representing the neighboring nodes of s by $nbrs(s)$, and the transition cost from a node s to another node s' as $c(s, s')$, the objective function $g(s)$ may be computed as follows:

$$g(s) = min_{s' \in nbrs(s)}(g(s') + c(s, s')) \quad (1)$$

The optimal path from s to the goal will then pass through the neighbor given by $s'' = argmin_{s' \in nbrs(s)}(g(s') + c(s, s'))$, and s'' is described as the parent of s'. On a regular two-dimensional grid, each node has eight neighbors, resulting in an eight-connected planning graph. For example, in Fig. 2, the state at location (x=1, y=2), which we will refer to as s_{12}, has neighbors s_{01}, s_{02}, s_{03}, s_{13}, s_{23}, s_{22}, s_{21}, and s_{11}. The figure shows objective function values that would be computed during eight-connected planning between the start state s_{23} and the goal state s_{00}, assuming a cell traversal cost of 100 (or $100\sqrt{2}$ when crossing the cell diagonally), and also assuming traversal costs are cast to integers.

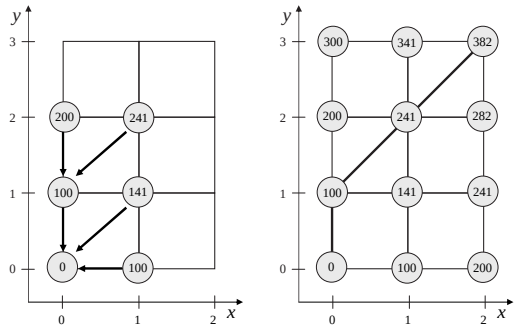

Fig. 2. Planning on an eight-connected graph. The computed objective function value of each node is shown. Left: A partial search showing the parent of each processed node. Right: The final computed path.

Three existing algorithms for continuous-field path planning on a regular grid are Field D* [2], E* [5] and Theta* [6]. These algorithms work primarily by modifying the manner in which objective function values (Equation 1) are computed during planning.

In Field D* [2], the path from s to the goal is allowed to pass through not only any of its eight neighbors, but also through any point along the edges between these neighbors. This is supported by estimating the objective function value of a point on an edge between two nodes as a linear interpolation of the objective function values of the two nodes. The algorithm thus solves a minimization problem based on this linear interpolation assumption to find the optimal crossing point along the edge. For example, in Fig. 3, the computed minimum estimated path cost of the node s_{12} involves crossing the edge between s_{01} and s_{11} at the point (x=0.55, y=1). In this manner, Field D* achieves continuous-field path planning on an eight-connected grid with arbitrary positive costs. Field D* is efficient and gives good results in practice [2]. However, it relies on the linear interpolation approximation which works fine when planning in a two-dimensional state space, but, as will be explained in the following section, can result in infeasible paths when applied to a domain with path-dependent state variables such as time and energy.

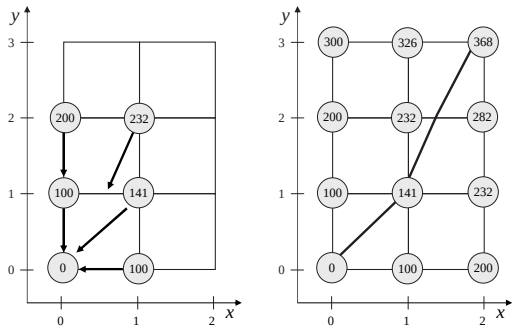

Fig. 3. Planning using linear interpolation in Field D*. Left: A partial search showing the parent of each node. Right: The final computed path.

E* [5] also uses interpolation to compute the objective function of nodes during the search, but it makes the interpolation method user-configurable, rather than restricted to linear interpolation, through the definition of a kernel function. Like in Field D*, the interpolation approximation can result in infeasible paths when applied to a domain with path-dependent state variables. This same argument applies to other approaches such as Konolige's Gradient Method [7] which apply interpolation to compute the path after costs have been propagated through the eight-connected grid.

In Theta* [6], continuous-field path planning is achieved by allowing the parent of a node s to be a node other than a local neighbor. In particular, when computing the objective function for a node s, the algorithm considers both the best path through a local neighbor, s', as given by Equation 1 and the path through the parent of a local neighbor, $parent(s')$, assuming that node s has line of sight to $parent(s')$–that is, assuming that the straight line between s and $parent(s')$ is not blocked by obstacle cells. Although Theta* is an exact algorithm and so does not have the same approximation-related problems as the interpolation-based approaches when applied to a domain with path-dependent state variables, it works only in an environment with binary costs, and no method currently exists for extending this algorithm to work with arbitrary positive cell traversal costs.

The TEMPEST/ISE global path planner [4] computes optimal eight-connected paths in a multi-dimensional state space including path dependent state variables such as time and energy. TEMPEST is the planner that models energy usage and the energy charging and storage functions of the battery, and reasons about terrain traversability. TEMPEST invokes the Incremental Search Engine (ISE) which employs heuristic search to find optimal eight connected paths subject to global constraints. ISE does not support continuous-field planning.

Existing algorithms for efficient replanning when changes occur in a graph include the D* family of algorithms (comprising D* [3], Delayed D* [8], D* Lite [9], Field D* [2], E* [5], DD* Lite [10] and others) and Adaptive A* [11]. In the former family of algorithms, the number of nodes processed during replanning is limited by repairing only the relevant parts of the search tree. In the last algorithm, it is limited by using the old search to determine more focussed heuristics for existing nodes. Adaptive A* is applicable only when costs can increase, not decrease. With the exception of Field D*, these re-planning algorithms do not directly support continuous-field planning. Furthermore, only DD* Lite [10] supports efficient planning with constrained path-dependent state variables by exploiting dominance relationships to improve the search efficiency.

Although each of the algorithms mentioned above addresses a subset of the capabilities required, none of them address all three capabilities. There is, as such, a need for a new approach to achieve continuous-field path planning and efficient replanning with constrained path-dependent state variables.

III. PROBLEM DESCRIPTION

The problem addressed in this work is that of efficiently searching a multi-dimensional space which includes path-dependent state variables, for a path that satisfies any constraints on the path-dependent state variables and that is not restricted to pass through only cell corners. More generally, when there are more than two independent parameters and hence the cell is greater than two-dimensional, we can say the the path is not constrained to pass through any resolution-independent subset of the cell boundary. Instances of this problem arise in several domains, such as when planning for a mobile solar-powered rover with limited battery capacity.

For a given domain of interest, we need to determine the independent and dependent parameters that constitute the configuration space of the robot. For example, the configuration space of the solar powered robot described earlier may include the spatial variables x and y as independent parameters, and the time and available energy at a given position as dependent parameters. Similarly, the configuration space of the data collection robot may include x and y as independent parameters, and time and storage space

as dependent parameters. Note that the state spaces under consideration in this problem are truly multi-dimensional in that the search process typically encounters several states with the same independent parameters and different dependent parameters.

Three distinguishing characteristics of dependent parameters are that (i) their values are path-dependent, (ii) their values are not known ahead of time but are computed during the planning process as states are being instantiated, and (iii) they employ dominance relationships. The first characteristic has already been explained. The second characteristic follows directly from the first because we cannot compute the path-dependent parameters of node until we have computed a path to that node, or equivalently to use the terminology of the previous section, until we know its parent node. Concerning the third characteristic, given two states in a search algorithm, s_i and s_j, a dominance relation D is defined as a binary relation such that $s_i D s_j$, that is, s_i dominates s_j, implies that s_j cannot lead to a solution better than the best obtainable from s_i [12]. Dominated states may be deleted without expansion in the search, thus eliminating entire branches of the search tree. An example of a dominance relationship involving a path-dependent state variable is the notion that, all other parameters being equal, a state with a lower energy requirement to reach the goal is always better than, and hence dominates, a state with a higher energy requirement to reach the same goal.

These characteristics of dependent parameters are at the root of the difficulty with applying an interpolation-based approach, such as Field D*, to this problem. Linear interpolation is acceptable for the objective function because the error in the approximation results in slight sub-optimality in the worst case. But it is unacceptable for constraint functions, which are defined in terms of dependent parameters, because the slight error introduced by the interpolation can lead to infeasibility.

In the example in Fig. 3, a path is planned from the start state s_{23} to the goal state s_{00} in a uniform cost field. The objective function value of s_{23} is computed, using interpolation, to be 368, and the corresponding optimal path is shown in the figure. Suppose the energy required to reach the goal is a path-dependent parameter of states in this space and the energy cost map for this problem is as shown in Fig. 4. In this map, the shaded cell has a higher energy cost (20) than other cells, where energy costs are 10. Using linear interpolation, the estimated value of the energy required at the crossing point (x=1.42, y=2) is 31. The distance from the crossing point to s_{23} is 1.15, and the cell's energy cost is 10. As such, we compute the energy requirement at s_{23} to be $\lfloor 31 + 10(1.15) \rfloor = 42$. If we extract the actual path from the optimal crossing point to the goal, however, we find that the true energy requirement at that point is 35, resulting in an energy requirement at s_{23} of 46. We have thus under-estimated the dependent parameter of s_{23}, and if the constraints in this domain capped the energy requirement at say 45, we would have an infeasible solution. This illustrates why linear interpolation cannot be used for this problem. Note that for clarity and due to the difficulty in graphically illustrating the three-dimensional state space, we show only the relevant state at each two-dimensional position. We will maintain this convention of using a two-dimensional representation of a higher dimensional space throughout the paper.

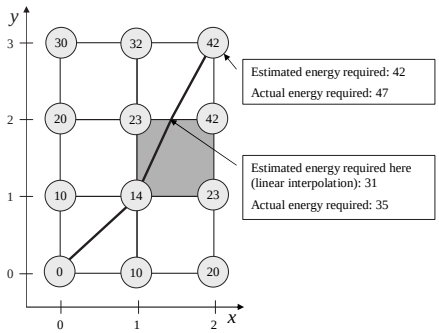

Fig. 4. The energy cost field and computed dependent parameters for the example in Fig. 3. The shaded cell has a higher energy cost.

IV. APPROACH

The key problem with the interpolation approach is the approximation of the dependent parameters of a state. To avoid this problem, we need to be able to compute exact values of the dependent parameters during planning. Since the values of the dependent parameters of a state s are computed from the dependent parameters of its parent state on the path from s to the goal, this implies that when s is instantiated, all states along the current best path from s to the goal must already be instantiated. That is, the path from s to the goal must pass through one of the neighbors of s. Combining this requirement with the idea that paths should be allowed to pass through the boundary of a cell at points other than the corner, means that during planning we must instantiate states along the boundary of the cell, resulting in the *framed cells approach*, illustrated in Fig. 5. This idea was first introduced as a way to plan better paths through quad-trees for cost minimization problems without dependent parameters or constraints [13], [14].

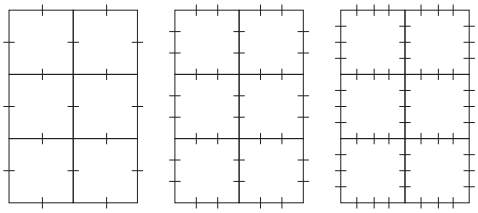

Fig. 5. The framed cells concept, showing subcell resolutions of two (left), three (middle) and four (right).

In the framed cells approach, we define a number of discrete points along the boundary of the cell where the path may cross the cell boundary. The number of these

crossing points is defined as the *subcell resolution*. Using a subcell resolution of one, the path may only transition through cell corners. With a subcell resolution of two, the path may additionally transition through the half-way point on the cell boundary, with a subcell resolution of 3, the path may transition through two additional points on the cell boundary, and so on. During the search, we instantiate states at these additional points along the cell boundary, effectively increasing the number of neighbors of a state and hence the number of allowed heading angles. After planning, we extract the path by following the gradient of objective function values from the start to the goal. An incidental advantage of the framed-cell approach is that it permits planning with complex objective/constraint functions, since we do not need to solve an optimization problem to compute the optimal crossing point on a cell boundary.

Fig. 6 illustrates the result of planning in a uniform cost field with a subcell resolution of three. Although states throughout the space would be expanded during planning, only states along the optimal path to the goal are shown. The image on the left shows the traversability cost field with the computed objective function value for each state. On the right, we have the energy cost field with the computed dependent parameter (energy required to reach the goal) for each state.

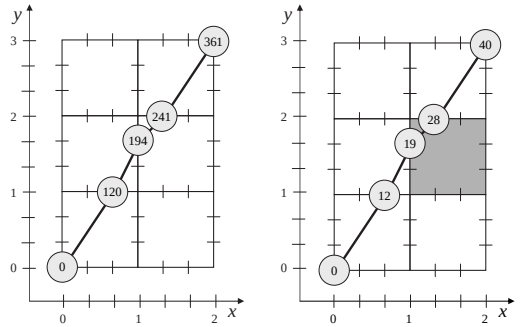

Fig. 6. Framed cells planning with a subcell resolution of 3. Left: the traversability cost field, showing computed objective function values of states. Right: the energy cost field, computed dependent parameter values of states.

The implementation of the framed cells approach to path planning uses an optimized version of the the DD* Lite algorithm [10]. DD* Lite is an incremental search algorithm that supports reasoning about state dominance and that enables rapid replanning by repairing only the relevant parts of the search tree when problem changes occur. To use DD* Lite for planning, we define state transition functions to compute the independent and dependent parameters of the predecessors and successors of a state, based on the framed cells configuration. These state transition functions require a reasonable model of the cost and constraints in the system. DD* Lite also requires a definition of dominance neighbors and the dominance relationship, both of which are defined in terms of the path-dependent parameters. We define the dominance neighbors of a state s as the set of states that share the same independent parameters but have different dependent parameters. Two types of dominance relationships are defined: true dominance, and resolution equivalency. The definition of true dominance depends on the particular problem domain of interest. For example, a state with a lower energy requirement to reach the goal may dominate a state with a higher energy requirement to reach the goal. Because the planning occurs in a multi-dimensional space, resolution equivalency is used to ensure the tractability of the search by defining dependent parameter bins within which states are considered equivalent. For example, a state with time 10:35am may be considered resolution equivalent to one with time 10:37am, and as such, only one of these states needs to be retained during the search.

Consider again the example of an energy-constrained exploration rover. If the energy costs are space dependent and not time dependent, we can plan in a three dimensional space consisting of two spatial dimensions, x and y, as independent state variables, and energy as a dependent state variable. The robot has a finite battery capacity, representing the energy available for it to reach the goal. The objective is to minimize traversal costs while satisfying energy constraints. We define state transition functions that use the framed cells framework to determine the x and y coordinates of the neighbors of a state and compute the energy coordinate based on the energy consumption model of the system. Fig. 7 shows an example environment with random traversal costs (left map) and high energy costs in the middle of the environment (right map). The planner is configured to use a subcell resolution of 4. The robot has a large battery capacity and starts out fully charged and this enables it to take the near-optimal path shown in the figure. Fig. 8 shows the same cost fields, but this time the robot starts off with less energy and cannot make it across the high energy cost region. The planner thus determines the least cost path that satisfies the energy constraints.

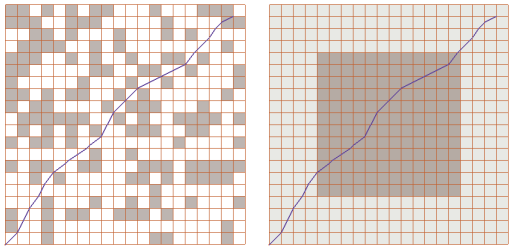

Fig. 7. A path planned with a subcell resolution of 4, for a robot with a large battery capacity, across an environment with randomized traversability costs and a high energy cost region at its center. The traversability cost map is shown on the left and the energy cost map on the right.

If in the above example the energy costs were time dependent as well as space dependent, time would need to be an added dimension, resulting in a four-dimensional state space. This is true for a solar powered rover because energy costs, which are considered negative for solar charging operations, depend on the time of day. In this case, the state transition functions may use information about the rover speed, energy

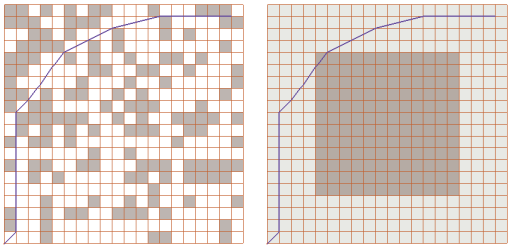

Fig. 8. A path planned with a subcell resolution of 4, for a robot with a small battery capacity, across an environment with randomized traversability costs and a high energy cost region at its center. The traversability cost map is shown on the left and the energy cost map on the right.

consumption model, solar charging model, time of day, and available solar flux to compute the time and energy coordinates of the neighbors of a state.

Note that the framed cells approach is not the same as simply using a finer grid for eight connected planning. Fig. 9 illustrates this by comparing, for a uniform cost field, the optimal eight-connected path on a coarse grid (left), the optimal eight-connected path on a grid of three times higher resolution (middle), and the optimal framed cells path with a subcell resolution of three (right). For the eight-connected paths, several equivalent optimal paths exist and may be computed by the planner, but we show only the path that looks most similar to the true optimal path. The figures illustrate that the eight connected paths in the coarse and fine grid are the same length, while the framed cells path is shorter and smoother because it allows a greater range of path headings.

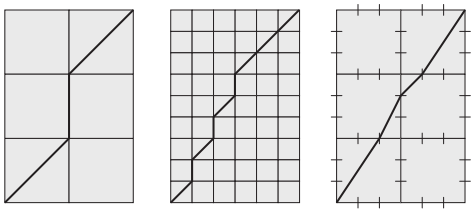

Fig. 9. Eight connected planning at a coarse resolution (left), eight connected planning at a fine resolution (middle), framed cells planning (right).

With the framed cells approach, we have increased both the number of states and the interconnectivity (number of neighbors for each state). Considering only a two-dimensional projection of the state space (ignoring dependent variables), an eight-connected graph derived from a regular grid with the length of the x dimension L_x and the length of the y dimension L_y has approximately $L_x L_y$ nodes and eight directed edges per node. With a subcell resolution of r, this graph will now have approximately $L_x L_y + (r-1)(L_x + L_y)$ nodes and $12r-4$ directed edges per node. This is illustrated in Fig. 10 for a subcell resolution of three. Because of this increase in complexity. planning time can be significantly longer for higher subcell resolutions than for eight-connected planning. To combat this, we need good heuristics to focus the search. In particular, we have been successful with using a two-dimensional search on the traversability cost map as a heuristic for the higher dimensional search, which is typically three or four dimensional in our problem domains of interest. The following section describes our computational results.

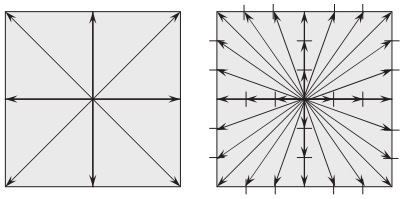

Fig. 10. Graph edges from a node at the corner of four cells: in eight connected planning (left), in the framed cells framework, with a subcell resolution of 3 (right).

V. RESULTS

We have tested our approach in simulation and on two mobile robots employing different state transition models.

Table I and Fig. 12 summarize the average performance of the planner in simulation on 20 different 50x50 maps with random traversability and energy costs. The traversability costs were strictly positive, while energy costs could be positive or negative, with negative costs representing solar charging. For these tests, the objective was to minimize traversal costs while satisfying energy constraints. Planning took place in a three dimensional state space consisting of x, y, and energy. To test replanning, modifications were made to both traversal and energy costs in a 3x3 region at the start location of the robot. Since this algorithm is suited for long-range navigation, it is often used with grid cells on the scale of meters or tens of meters, and is used in conjunction with a local navigator for small-scale obstacle avoidance. Because of the size of the grid cells, the robot is likely to discover changes in a small number of cells near its current location, hence the choice of the 3x3 modification region. The results are shown for three different scenarios, termed “loose energy constraints”, “moderate energy constraints” and “tight energy constraints” respectively. In the first scenario, the simulated robot’s battery capacity was large enough that, when fully charged, it could take the optimal path (with respect to traversal costs) from the start to the goal without being constrained by energy. In the second scenario, the robot’s battery capacity was such that its path was on average 1% longer than optimal due to detours as a result of energy constraints. In the third scenario, the robot’s battery capacity was such that its path was on average 6% longer than optimal due to energy constraints. Table I shows the average path cost for initial planning as well as after replanning for subcell resolutions of 1 (equivalent to eight-connected planning), 2, and 3. The path costs are shown relative to the eight-connected path cost. Although the relative savings in path cost are fairly small for these randomized cost fields, the real benefit of continuous field path planning is in the smoothness

	Sub-cell resolution		
	1	2	3
Loose energy constraints			
Relative path cost	1.000	0.979	0.975
Relative path cost after replanning	1.000	0.979	0.975
Moderate energy constraints			
Relative path cost	1.000	0.980	0.975
Relative path cost after replanning	1.000	0.980	0.975
Tight energy constraints			
Relative path cost	1.000	0.976	0.970
Relative path cost after replanning	1.000	0.975	0.970

TABLE I

PATH COST RATIOS WITH THE FRAMED CELLS IMPLEMENTATION

of the path, as demonstrated by the example in Fig. 11, and by the robot tests described later in this section.

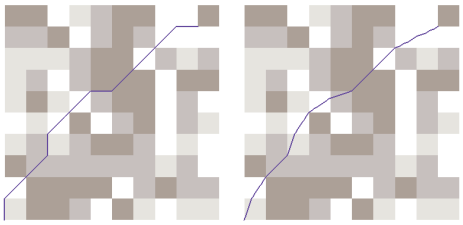

Fig. 11. Comparison between an eight-connected path (left) and framed cells path with subcell resolution of 3 (right), in a random cost field.

Fig. 12 shows the average planning and replanning time for subcell resolutions of 1, 2, and 3 and for the three different energy scenarios described above. In these test cases there was a single goal state, although the algorithm can handle multiple goals. The replanning times shown are the sum of the map update time and the actual replanning time, on a Pentium M 770 2.13 GHz processor. Results are shown for planning and replanning using the Euclidean distance heuristic as well as the two dimensional search heuristic described in the previous section. The first trend to notice is the price that is paid in planning time for the smoother, more optimal paths that are the result of higher subcell resolutions. However, the two dimensional search heuristic focusses the search so much better than the Euclidean distance heuristic does that this significantly reduces the planning time to reasonable values, even for a subcell resolution of 3. With the Euclidean distance heuristic, replanning is significantly more efficient than planning from scratch after a map modification occurs. With the two dimensional search heuristic, replanning can be slightly less efficient than planning from scratch. This is because when traversability costs change, the heuristic value of a state may change since the heuristic value is computed as a two-dimensional search of the traversability cost map. As such, the heuristics need to be recomputed for each state on the open list when map modifications occur, and the list re-ordered accordingly. This negatively impacts the replanning time. Another pattern to notice in the results is that the planning time when using the Euclidean distance heuristic reduces with a smaller battery capacity of the robot (tighter energy constraints). This is simply due to a reduction in the size of the three-dimensional space to be searched. This reduction in planning time is not obvious when using the two dimensional search heuristic because the three-dimensional search is very focused in this case.

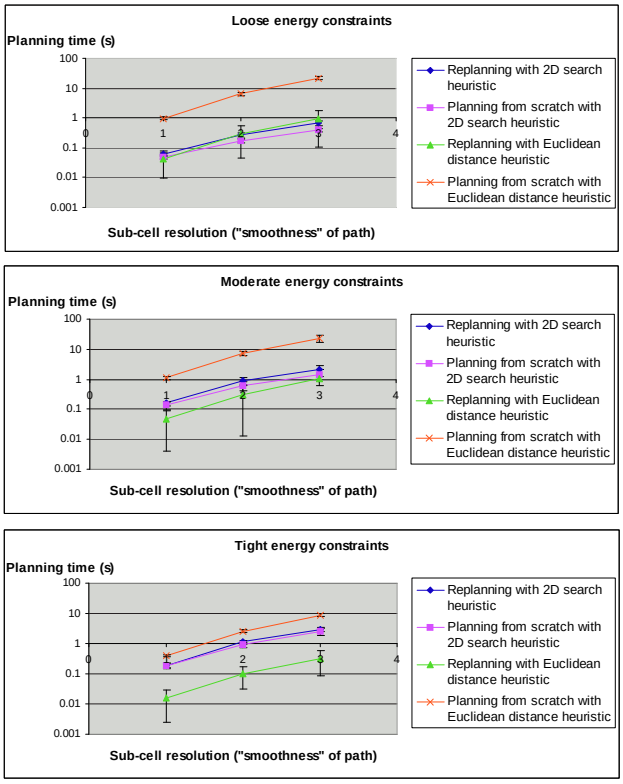

Fig. 12. Planning and replanning times with the framed cells implementation (in simulation)

We compared the planning and replanning times for the framed cells approach with those for a higher resolution eight-connected grid, for twenty 50x50 maps with random cost fields and moderate energy constraints as described previously. The results are shown in Fig. 13 for five planning instances: basic eight-connected planning, framed cells planning with a subcell resolution of two, eight connected planning with a twice higher resolution, framed cells planning with a subcell resolution of three, and eight-connected planning with a three times higher resolution. When using the Euclidean distance heuristic, the planning and replanning time is higher for the framed cells approach than for a higher resolution grid. This can be explained by the higher branching factor in the framed cells implementation. This is also true, although to a lesser extent, when replanning with the two-dimensional search heuristic. However, when planning from scratch with the two-dimensional search heuristic, the planning is quicker with the framed cells approach than with the higher resolution grid. This is because the highly focused nature of the search compensates for the increased branching factor. It should be noted that as previously described, using a higher resolution grid does not solve the fundamental problem associated with eight-connected planning, which is the limited number of robot headings.

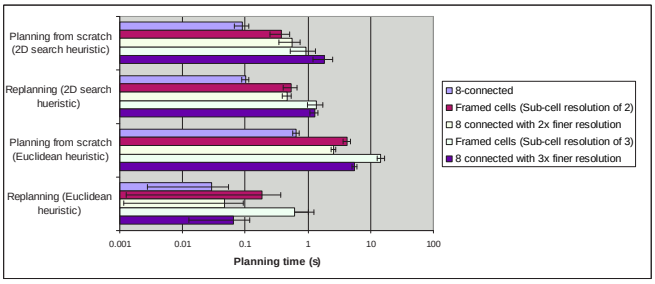

Fig. 13. Comparison of planning and replanning times for the framed cells approach and grids of various resolutions

The framed cells path planning algorithm was tested with an ActivMedia Pioneer 2-DX robot [15] in an indoor environment with simulated energy constraints. Planning was done off-board and the plan waypoints were communicated wirelessly to the robot. The robot, shown on the left of Fig. 14, used encoders and a gyroscope for dead-reckoning and localization, and a SICK scanning laser range-finder for obstacle detection. The first set of tests required the robot to traverse an open environment with uniform traversal costs and a simulated high energy cost region at its center, illustrated in Fig. 15. The objective was to optimize traversal costs while satisfying energy constraints. The tests were run with a subcell resolution of 1 (eight-connected planning) and a subcell resolution of 4. With a high simulated battery capacity, the robot's path took it directly to the goal, across the high energy cost region at the center of the environment. The path computed with a subcell resolution of 4, shown on the right side of Fig. 15, was significantly smoother and more efficient than the eight-connected path, shown on the left. In these figures, the planned path is shown with a dark solid line and the robot's path with a lighter dotted line. The robot's path was captured via position updates published by its localization system.

Fig. 14. The robots used for testing. Left: The ActiveMedia Pioneer 2-DX robot. Right: The solar powered rover

With a low simulated battery capacity, the robot's path avoided much of the high energy cost region at the center of the map, as shown in Fig. 16. Again, the path computed with a subcell resolution of 4, shown on the right, is smoother than the eight-connected path, shown on the left.

The second set of tests with the Pioneer robot again involved an environment with uniform traversal costs. The energy costs were mostly uniform, but with a handful of

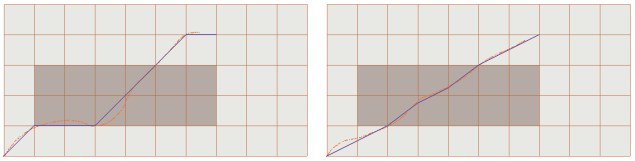

Fig. 15. A path executed on the Pioneer robot assuming a large battery capacity, across an environment with uniform traversability costs and a high energy cost region at its center, Left: Using a subcell resolution of 1 (eight-connected). Right: Using a subcell resolution of 4.

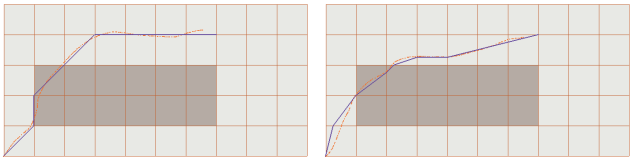

Fig. 16. A path executed on the Pioneer robot assuming a small battery capacity, across an environment with uniform traversability costs and a high energy cost region at its center, Left: Using a subcell resolution of 1 (eight-connected). Right: Using a subcell resolution of 4.

negative energy cost "charging" cells scattered in the environment, as shown in Fig. 17. With a high battery capacity, the robot was again able to make it directly from the start to the goal (this path is not shown). With a low battery capacity, however, the path made a detour through a couple of the charging cells in order to satisfy energy constraints. Again, the path computed with a subcell resolution of 4, shown on the right, is smoother than the eight-connected path, shown on the left.

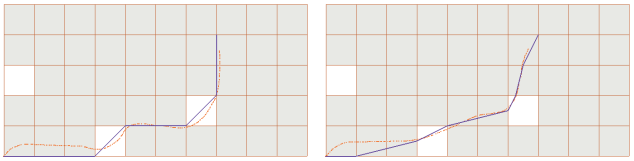

Fig. 17. A path executed on the Pioneer robot assuming a small battery capacity, across an environment with uniform traversability costs and isolated negative energy cost ("charging") cells. Left: Using a subcell resolution of 1 (eight-connected). Right: Using a subcell resolution of 4.

Finally, we tested the algorithm on a solar powered rover shown on the right in Fig. 14. The rover uses two cameras for terrain evaluation during navigation and utilizes wheel encoders and a gyro for motion estimation. Its sophisticated software environment includes a rover executive, a mission/global path planner, and a local navigator. For these experiments, we integrated the framed cells planner with the TEMPEST mission planning software [4], replacing the Incremental Search Engine (ISE) [4] which was previously used to compute paths. This enabled the state transition functions implemented for the framed cells planner to take advantage of the world, rover, and lighting models defined by TEMPEST. Planning, which was done onboard the rover, used a four dimensional space with the spatial variables x and y as independent parameters, and time and energy

as dependent parameters. Time was included in the state space because energy costs are time-varying due to the time-varying nature of available solar flux for charging the robot's batteries. The objective of the plan was to minimize path traversal time while satisfying energy constraints as well as maximum slope constraints. Fig. 18 shows plans generated by the TEMPEST/framed cells planner combination on a 10m resolution map, during field tests at Amboy Crater, CA. The plan at the top left was generated using a subcell resolution of 1 (eight-connected planning), while the significantly smoother path at the top right was generated with a subcell resolution of 3. The rover navigated around large-scale terrain features by following waypoints from these plans generated by the high-level TEMPEST/framed cells planner, while avoidance of local sub-meter scale obstacles such as rocks and shrubs was ensured by the lower-level navigator software. In the the bottom left and right of the figure, we show the actual paths taken by the rover while following the eight-connected and smoother plans respectively. The rover path was captured by subscribing to messages published by the robot's state estimation (localization) system. For the eight-connected run, the rover traveled 324m in 1605 seconds while the length of the smoother run with a subcell resolution of 3 was only 290m. Furthermore, traversing the smoother path took only 895 seconds. This represents a 10% improvement in distance and 44% improvement in time for continuous field path planning over eight-connected planning. The significant time reduction for the smoother plan was caused in part by a reduced average rover speed for the eight connected plan, as the local navigator encountered small-scale obstacles in the environment. In all, our initial field tests at Amboy Crater involved 2.15 km of autonomous driving using the TEMPEST/framed cells planner combination as the high level path planner.

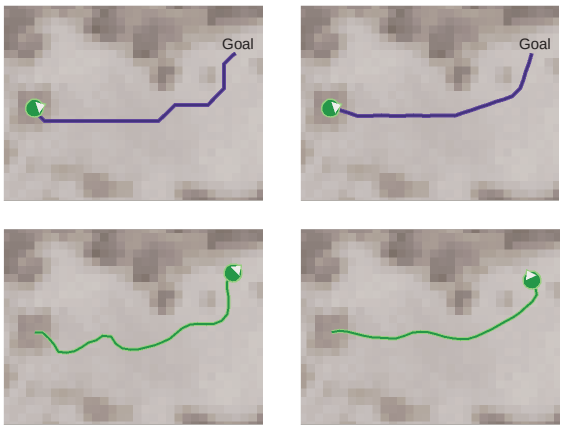

Fig. 18. Example run of the TEMPEST and framed cells planner combination. Top left: Planned path using a subcell resolution of 1 (eight-connected). Top right: Planned path using a subcell resolution of 3. Bottom left: Actual rover path with a subcell resolution of 1. Bottom right: Actual rover path with a subcell resolution of 3.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we present our work on continuous-field path planning with path-dependent state variables. We formulate a framed cells approach for continuous-field path planning and combine this with incremental heuristic search and reasoning about dominance to create an effective algorithm for a previously unsolved problem. We have tested this algorithm in simulation and on two different robots and demonstrated that it is capable of computing smoother, more efficient plans for a mobile robot with path-dependent state variables. The major drawback of the presented approach is the increase in planning time as we strive for smoother paths by using higher subcell resolutions. However, by using a two-dimensional search heuristic to guide the higher dimensional search, we obtain tractable planning times for reasonably-sized problems.

Future work includes achieving further improvements in planning time and memory utilization with this planning approach. It will also be interesting to explore ways to automate the decision of the best subcell resolution to use for a given problem. Finally, we will continue to field-test the planner on the robots.

VII. ACKNOWLEDGMENTS

This work was sponsored by the Jet Propulsion Laboratory, under contract "Reliable and Efficient Long-Range Autonomous Rover Navigation" (contract number 1263676, task order number NM0710764). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the U.S. Government.

The authors wish to acknowledge the invaluable support of Marc Zink, David Wettergreen, Dominic Jonak, David Thompson, and Francisco Calderon in field testing this work. They are also appreciative of Marc Zink's contributions in creating visualization tools for the planner as well as libraries to interface with the Pioneer robots.

REFERENCES

[1] "Mars Science Laboratory," http://mpfwww.jpl.nasa.gov/msl/.

[2] D. Ferguson and A. T. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, October 2005.

[3] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '94)*, vol. 4, May 1994, pp. 3310 – 3317.

[4] P. Tompkins, A. T. Stentz, and D. Wettergreen, "Mission-level path planning and re-planning for rover exploration," *Robotics and Autonomous Systems, Intelligent Autonomous Systems*, vol. 54, no. 2, pp. 174 – 183, February 2006.

[5] R. Philippsen, S. Kolski, K. Macek, and R. Siegwart, "Path planning, replanning, and execution for autonomous driving in urban and offroad environments," in *In Proceedings of the Workshop on Planning, Perception and Navigation for Intelligent Vehicles, ICRA*, Rome, Italy, 2007.

[6] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007, pp. 1177–1183.

[7] K. Konolige, "A gradient method for realtime robot control," vol. 1, 2000, pp. 639–646 vol.1.

[8] D. Ferguson and A. T. Stentz, "The delayed D* algorithm for efficient path replanning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005, pp. 2045 – 2050.

[9] S. Koenig and M. Likhachev, "D*lite." in *AAAI/IAAI*, 2002, pp. 476–483.

[10] G. A. Mills-Tettey, A. T. Stentz, and M. B. Dias, "DD* lite: Efficient incremental search with state dominance," in *Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, July 2006, pp. 1032–1038.

[11] S. Koenig and M. Likhachev, "Real-time adaptive A*," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006, pp. 281–288.

[12] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*. Computer Science Press, 1978.

[13] D. Z. Chen, R. J. Szczerba, and J. J. Uhran Jr., "Planning conditional shortest paths through an unknown environment: a framed-quadtree approach," in *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots'.*, vol. 3, 1995, pp. 33–38 vol.3.

[14] A. Yahja, S. Singh, and A. T. Stentz, "An efficient on-line path planner for outdoor mobile robots operating in vast environments," *Robotics and Autonomous Systems*, vol. 33, no. 2 and 3, pp. 129–143, August 2000.

[15] "Pioneer P3-DX robot," http://www.activrobots.com/ROBOTS/p2dx.html.