Adaptive Anytime Motion Planning For Robust Robot Navigation In Natural Environments

Mihail Pivtoraiko
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
Email: mihail@cs.cmu.edu

Abstract— **The problem of robot navigation is treated under constraints of limited perception horizon in complex, cluttered, natural environments. We propose a solution based on our previous work in fast constrained motion planning, where arbitrary mobility constraints could be satisfied while the planning problem is reduced to unconstrained heuristic search in state lattices. By trading off optimality, we improve planner run-times and increase robustness through achieving anytime planning quality, such that it becomes possible to integrate the planner within the high speed navigation framework. We show that using a planner in navigation works well and fast enough for real vehicle implementation, while it presents a number of important benefits over state-of-the-art in navigation.**

I. Introduction

Natural environments pose special challenges for autonomous robots. Such environments are typically unstructured, sometimes quite cluttered, and unknown ahead of time. An autonomous mobile robot traversing such environments must be able to detect and avoid a wide variety of obstacles. We define the basic problem of robot navigation as that of enabling a mobile rover to traverse very long distances autonomously and efficiently. In accomplishing this task, we assume a robot perception system that produces accurate results only within a limited radius in front of the vehicle. This perception information is assumed to be updated quickly enough such that the vehicle can stop when an obstacle is detected in front.

Solving this problem is valuable as robust autonomy of mobile robots results in their increased utility. The problem of cross-country navigation is difficult for several important reasons, in particular the lack of knowledge of the terrain, and the desire to traverse it at as high speed as possible. The intelligent navigation solution must be able to command high vehicle velocity in smooth terrain, and slow down accordingly when rough terrain is encountered. Lastly, many applications especially benefit from the use of wheeled vehicles, and the exploration goals must be accomplished by the navigator under the kinematic constraints of such vehicles. We offer a solution that effectively satisfies all these requirements.

The problem we consider has received considerable attention. Most well-known solutions include the methods of navigation that are based on the evaluation of constant-curvature arcs ([5], [10], [20] and others). These methods have been used extensively in field work and proved to be reliable and computationally efficient. Henceforth we refer to them as *arc-based navigators*. Most of the inspiration for this work came from field experience with these navigators, and we propose our method as an incremental improvement over them. In particular, we preserve a paradigm of using a global planner to guide the vehicle in the general direction of the goal, while a local process makes decisions for nearby motions. The key innovation is using a full vehicle-aware motion planner instead of the arcs and making it run fast enough such that the paths could be replanned continuously. With this inhancement, we propose imparting the vehicle an ability to look further ahead and simulate motions beyond its immediate neighborhood (although still within the perception horizon), including the area where the vehicle already travelled. The exploration of this greater area is accomplished via heuristic search using a set of motion primitives, conceptually similar to the discrete immediate motions used in arcs-based methods. This allows the vehicle to make intelligent decisions while traversing difficult terrain with limited perception, and the necessary maneuvering (which may including backing up) can come out naturally.

In the balance of the paper we explain many benefits of such an enhancement to current navigation solutions, discuss how modern computational hardware makes this feasible for real-time execution, and present compelling simulation results that demonstrate the effectiveness of our method.

II. Prior Works

This work may be viewed as a synthesis of our previous research in motion planning, field experience and successfull ideas of others. There is a significant heritage in outdoor navigation for mobile robots.

In the nineties, it became possible to perform interleaved planning and execution, and achieve higher speeds. For example, the AVL and DEMO vehicles achieved notable results in cross-country navigation by developing the algorithms that continuously evaluated perception information and simplified the planning problem to a manageable level by only making the immediate decisions for the robot. By running this algorithm in a tight loop, sufficient response time was achieved that the vehicle could safely travel continuously at reasonable speeds for navigation. The aforementioned arc-based navigators ([5], [10], [20], etc.) evaluate a set of predetermined arcs

978-0-7695-3753-5/09 $25.00 © 2009 IEEE
DOI 10.1109/AT-EQUAL.2009.33

against available perception information and utilized an arbiter module to pick the best arc.

Using planning via search has been suggested as an important advantage, and [12] presented a successfull planner that achieved efficiency by operating in a limited area in front of the vehicle. Previously full state-space search has been prohibitively expensive in the framework of navigation, as suggested in [10], however thanks to the latest advances in this technology, its applications to this problem became feasible, as we attempt to show here. In [18] we have demonstrated planners based on search in state lattices that are quite efficient. Here we build on that work and present a number of significant modifications that are necessary to obtain robustness and performance guarantees of these planners to be used in high-speed navigation.

In order to achieve planning robustness, which is paramount in real-time navigation applications, we build on the body of literature in time-critical and anytime planning. Important inspiration for this work came from the results in [14], [15], [21], among others. We incorporate the analysis and results in [1] regarding sub-optimal search using overestimating heuristics, while enforcing a bound on sub-optimality [15]. To our knowledge, this is the first work, apart from others in our lab, that achieves robust navigation at the similar or better runtime efficiency than the state-of-the-art in navigation.

III. Fast Constrained Motion Planning via State Lattices

In this section we offer a brief introduction to our recent work ([18], [19]) on efficient motion planning under differential constraints. It is based on identifying a minimal set of motion primitives and using them within simple unconstrained search, such that the solution is an inherently feasible path. In this discussion, we are primarily interested in wheeled vehicles moving on rough unknown terrain. We do not constrain the type of vehicles: an arbitrary system model formulation can be used (including car-like and n-trailer vehicles).

A. State Lattice

Discrete representation of states is a well-established method of reducing the computational complexity of planning at the expense of reducing completeness. However, in motion planning, such discrete representations complicate the satisfaction of differential constraints which reflect the limited maneuverability of many real vehicles. We propose a mechanism to achieve the computational advantages of discretization while satisfying motion constraints.

To this end we have previously introduced a search space, referred to as the *state lattice*, which is the conceptual construct that is used to formulate a nonholonomic motion planning query as graph search. The state lattice is a discretized set of all reachable configurations of the system. It is constructed by discretizing the state space into a hyperdimensional grid and attempting to connect the origin with every node of the grid using a feasible path, an edge, by utilizing an inverse trajectory generator.

1) State Discretization and Regularity: Discretization converts the motion planning problem into a sequential decision process. We adopt the typical strategy of assuming that decisions are made only at discrete states. While the state vector can certainly have arbitrary dimension, we previously have implemented the state lattice in 4 dimensions: each lattice node represents a 2D position, heading and curvature. In the following sections we propose an extension of this method that also considers translational and angular velocities as state variables.

If the discretization exhibits any degree of regularity, then the spatial relationships between two given states will reoccur often due to the existence of other identically arranged pairs of states. Such regular discretization leads to a set of motion options which is similarly regular. Through studying the regularity of all spatially distinct feasible motion alternatives, we are able to remove redundancy and arrive at the *minimal representation* of the overall collection of motion options.

2) Inverse Trajectory Generation: With these properties in mind we construct the state lattice by using the inverse path generator to find paths between any node in the grid and the arbitrarily chosen origin. By regularity, we can copy the resulting set of feasible paths to any node in the lattice. In the limit, as the lattice is built by including all feasible motions from any point, it will approach the reachability graph of the vehicle, up to a chosen resolution. As shown in [19], the state lattice can be considered a valid representation of the system's reachability graph.

There are a number of important benefits of using the inverse trajectory generator rather than the forward one.

- Regular coverage of the discretized state space is guaranteed. This is not easy to achieve by discretizing the controls and integrating model equations as done in [12], [13]. As shown in [16], finding a discretization in controls that results in a discretization in state space is possible, but it is usually difficult to achieve. Conversely, by simply using the inverse generator to create all paths of interst, we can choose a convenient discretization that yields most efficient planning.
- The research in efficient trajectory generation via optimal control has received considerable attention over many years, and currently an appreciable number of methods are available. The generator in [8] that we evaluated is able to compute trajectories in a few milliseconds. Important improvements over this work, in particular to handle rough terrain and actuator dynamics, have been introduced in [6].
- Since we build on existing technology in trajectory generation, we note that solutions such as [6] and [8] can generate trajectories for an arbitrary formulation of the system model.
- A vehicle model can be easily modified without changing the state space discretization. For example, we can add consideration of wheel slip, delays in steering and other actuator response only by updating the system model considered by the trajectory generator, while the rest of

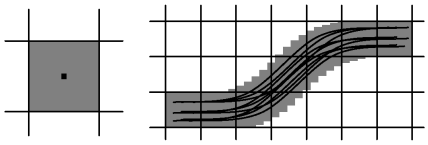

Fig. 1. Spatial equivalence classes for points and paths. All of space in a rectangloid in C-space is placed in an equivalence class represented by the point at the center. Similarly, all paths contained within the swath swept by a cell following a path can be placed in an equivalence class of paths.

the solution remains the same.

B. Primitive Control Set

The state lattice introduced so far can already be used as a search space for motion planning. However, the efficiency of the planning could be significantly improved by avoiding representing the entire lattice explicitly. Instead, we attempt to obtain a finite representation of the lattice. This is identical to capturing the local connectivity of the vehicle's reachability graph, i.e. the collection of paths in the near vicinity of the vehicle. In [19] we introduced a principled method of capturing this connectivity that both offers practical guarantees of best exploration of the lattice and limits itself to considering paths in a small local neighborhood to preserve search efficiency. This leads us to obtain the minimal set of controls as the best representation of all motion alternatives. Obtaining this control set is done through addressing two dilemmas: the inifinite paths in the generic lattice and the infinite density of them.

Since we discretize state space, it is consistent to consider discretizing paths through space in a similar fashion. We consider two paths (with identical endpoints) which are "sufficiently" close together to be *equivalent*. We define a path τ_1 to be equivalent to τ_2 if τ_1 is contained in a certain region Q around τ_2, defined as a set of configurations within a certain distance δ_e of τ_1, given some metric ρ:

$$\forall q \in \tau_1, \forall q' \in \tau_2, Q = \{q' | \rho(q', q) < \delta_e\} \qquad (1)$$

All paths that satisfy this criterion are considered to belong to the same equivalence class (Fig. 1). It is important to note that this definition of path equivalence is consistent with applications to mobile robotics. Typically, there is a certain error of path following for realistic vehicles. By allowing path equivalence, and hence path discretization, we exploit this error and obtain more efficient motion planning [18].

As for infinite extent of the paths, note that the longer the path is, the more likely there is a concatenation of path segments that is equivalent to the original path. An algorithm for obtaining the control set for a lattice, known as *path decomposition*, was presented in [18]. The present discussion is general w.r.t. the methods for generation of the motion primitives. Other ways of generating control sets are acceptable, as long as the paths end on cell centers, such that we can assume regularity in state space discretization.

C. Motion planning as search using control sets

Once the lattice has been created and its finite representation has been constructed off-line, motion planning can be formulated as basic search in this space. [18] showed A* applied to this problem and quoted impressive results, where the speed of nonholonomic planning approached, and in some cases, exceeded, the speed of basic 2D grid planning at the same resolution.

IV. Extention of Lattice Motion Planning for Navigation

The problem of navigation in unknown environments poses a number of additional requirements. In particular, perception is reliable only in a limited area around the vehicle. This imposes a bound on the quality of the decisions the vehicle can make. However, it is best to guarantee both efficient traversal of the terrain as well as the safety of the vehicle. In this section we discuss how the motion planner can be adapted to this problem. In particular, we note that the cost-to-goal heuristic estimate needs to account for the fact that the cost of traversal to the goal is not yet known. Also, the notion of the goal must be modified since non-holonomic planning all the way to the goal would be inefficient, especially since this plan would need to be modified as new perception information is acquired.

A. Considering Velocity

The optimal solution may include traversing the path at varying velocity, and vehicle dynamics come into play. Therefore, it is important to consider vehicle velocity during search. We extend our planner to handle velocities by repeating the same process for determining control sets at evenly discretized translational and angular velocities, v and ω, respectively. This is similar to [12], however it allows us to preserve the features of the state lattices. Using the inverse trajectory generator prevents paths from being dense in the workspace, and enables them to span it more efficiently. Considering primitives at several levels of v and ω increases the state space considerably, but due to specially designed search heuristics, the planner still operates very efficiently.

B. Node Cost

In order to calculate the cost of traversing lattice edges during planning, we perform convolution of the vehicle frame over the costmap that contains available perception information about the terrain. The primary pusrpose of this costmap is to indicate obstacles and other untraversable regions. Such regions are denoted in a special manner such that the planner will not generate paths that go through them. Besides, the costmap contains other regions that are allowed, but may be tougher to traverse for a variety of reasons, e.g. slopes, slippery areas, etc. All such regions are uniformly indicated as high-cost areas in the map and are considered uniformly by the planner.

However, convolution is a fairly computationally expensive procedure. To improve run-time, we exploit the technique derived from the method introduced in [12]. Each primitive

path in the control set is associated with a list of map cells that it covers, i.e. that correspond to the projection of the control primitive onto the workspace. This list is pre-computed off-line. Then, in order to estimate the cost of a particular primitive, we simply iterate over this list and sum the sampled cell costs starting at the vehicle's pose in the costmap. This process allows us to consider non-binary obstacles which is important for achieving a tunable trade-off between traversal time and the overall cost of traversal. Being able to satisfy this trade-off is especially important in off-road navigation scenarios. Moreover, the procedure of summing values over an array of data is a very efficient operation and can be greatly optimized: modern processors offer special instructions to accomplish this in hardware (e.g. SIMD extensions on Intel® or other processors).

C. Termination Condition

Since the motion planner executes within the navigation scenario, the actual goal of the traversal may be a long distance away, most of which lies over unknown environment. Hence, it is inefficient to execute the planner all the way to the goal. Unlike [14] and [15], we execute the planner in a small neighborhood of the robot where perception information is reliable. Typically real sensors have a certain distance where their readings are reliable, and this deffines a certain *perception horizon* radius, R_p, around the vehicle [9]. Besides the distance from the vehicle, the second criterion for the goal states is alignment with the general direction to the goal, as suggested by the D* cost. Since D* cost is typically available at all lattice nodes near the vehicle, it can be viewed as a discrete *field*. The gradient of this field indicates a general direction toward the goal. Thus, we define a *goal region* to be a subset of the state space S which includes the states whose translational coordinates are at least a specific distance $R \leq R_p$ to the robot position (using Euclidean metric) and whose heading θ_s is aligned within a certain threshold Δ_θ to the direction of the gradient of D* field, θ_{D^*}:

$$\forall s \in S, G = \{s | L_2(s, s_0) < R, |\theta_s - \theta_{D^*}| < \Delta_\theta\} \quad (2)$$

In the following sections we describe how the goal region can be modified to support anytime plannint properties. In particular, note that there is a relationship between the magnitude of distance threshold R and relative difficulty of planning: the smaller this value, the fewer search iterations are required to achieve a goal state $s \in G$.

D. Weighted Heuristic for Planning in Navigation

Similar to other work in this area, namely [14], we are using fast 2D search as heuristic. Since in the problem of navigation the environment is largely unknown, it is helpful to plan the overall path backwards: from the final goal to the vehicle. In this manner the root of the search tree can remain the same. Moreover, as new environment information is uncovered as the vehicle is moving, it is necessary to avoid the complete replanning. A good 2D search algorithm that satisfies these requirements is Field D* [3]. The weighted node costs of this algorithm are used as heuristic values.

In the navigation problem as described, the planning problem is quite complex and the available time for planning can be very scarse. Due to these constraints, generating optimal solutions often can be impossible. In these situations, the planner must be able to find the best solution that can be generated within the available time. Well-known algorithms that satisfy such requirements are known as anytime algorithms [2], [15], [21]. The anytime planning algorithms that are based on A* make use of the fact that in many domains inflating the A* heurisic values often provides significal improvement in search runtime at the cost of optimality [1], [11]. Besides, it was shown [17] that if the consistent heuristic is used and the heuristic values are multiplied by an inflation factor $\epsilon > 1$, the cost of the generated solution is guaranteed to be within a factor of ϵ of the cost of the optimal solution. In other words, ϵ serves as a bound on sub-optimality of the produced solution. This parameter allows us to to achieve reactive real-time execution times while ensuring a bound on the sub-optimality of the solution.

E. Anytime Planning

For using planning within a navigation framework that is executed on a vehicle that is moving at high speed, it is critical to have runtime guarantees. Anytime planning therefore is especially attractive for this application. Our formulation of the navigation solution allows achieving anytime planning in two ways. First, the solution lends itself well to leveraging the standard anytime planners. Through utilizing the state lattice, nonholonomic planning reduces to basic search in this specialized search space and therefore is "mechanically" identical to the grid search. It follows that anytime algorithms such as ARA* [15] can be readily used for this application.

A second way for adopting anytime ideas and improving robustness is via modifying the meaning of the goal by extending the goal radius R. This extension can be viewed as an incrementally increasing depth of search. After just one iteration of the search, the planner will have identified a path segment that is the bast way to go to avoid obstacles and move in the best direction toward the goal (as indicated by the summation of the cost of the segment and the heuristic, here based on the D* cost). If time allows, the planner is allowed to proceed with search to the next, greater, value of R. Thus the search continues incrementally farther away from the vehicle. The search will terminate when the goal region reaches the perception horizon. In terms of A* search, this means that when a goal state is about to be popped from the top of the OPEN list, we simply extend the goal region and continue the searching algorithm. In this way we reuse all the previous computation with no additional overhead, while the search can be interrupted and a valid path to a previous value of R is returned and is readily available for execution on the vehicle.

Thus, in easy environmetns the planner will be able to progress the search up to R_p quickly. In more complex

environments, the planner will only be able to plan very near the vehicle. The robot will still have paths to execute, but these delays can serve as cues to the robot that environment is complex. This will result in an intelligent behavior of slowing down. As the vehicle slows, the planner will have more time to find better plans (perhaps by additionally decreasing the suboptimality bound ϵ for better quality solutions), such that it can get out of tough spots with the least risk.

V. Heritage of Arcs-based Navigators

We would like to acknowledge once again that much of the inspiration for this work came from extensive field experience with arc-based navigators. Such algorithms have been successfully utilized over the years. In particular, our work on Perception for Off-road Robotics (PerceptOR) and Learning Applied to Ground Robots (LAGR) projects (supported by DARPA) enabled us to identify the strengths and ideas for improvement of these algorithms. Most of these ideas have been incorporated in this work. We hope that this work will serve as an incremental improvement over that proven technology.

A. The Similarities

It is important to point out that there are a number of structural parallels between arcs-based navigation algorithms and the motion planner introduced here. We outline most important ones below.

- Each node expansion of the planner's search algorithm (e.g. A* or ARA*) is functionally equivalent to arc generation in arc-based approaches. Note that the arcs are typically generated by forward simulation of the dynamic vehicle model, given current vehicle state. As described in Section III-B the present planner makes these primitives readily available, and some efficiency is gained in avoiding online execution of the mathematics for forward vehicle model simulation.
- Each evaluation of node cost during search is similar to "tactical" trajectory evaluation during each cycle of an arcs-based navigator. Often this step is also implemented by convolution of vehicle frame over the cost map, as we suggest in Section IV-B. We also point out that a typical number of different arcs that are evaluated is over a dozen [20], [5] which is greater than the average outdegree of the sets of motion primitives that were shown sufficient for capturing the variety of immediate motions of similar vehicles through the process of control set generation in [19]. One reason for this is that the control set primitives are certainly not constrained to constant curvature like the arcs are. This is a savings both in the degree of node expansion and in the amount of convolution computation that needs to be performed.
- Each evaluation of node heuristic during search is similar to the "strategic" trajectory evaluation during each cycle of arcs-based navigators. This evaluation is crucial to guiding the vehicle in the general direction toward the goal. For approaches that rely on D* or similar global search methods, this amounts to reading the cost to goal from the terminal points of each arc. In our approach we use the same approach in assigning heuristic values to nodes.
- The conclusion of each cycle of typical arcs-based navigators is execution of an arbiter that combines both "tactical" and "strategic" scores into a single measure which dictates which arc is chosen for execution. There has been a fair amount of research on best ways to do this ([20] and others). The same basic question is important in heuristic search, in combining the cost and heuristic value of a node into its overall cost (e.g. *f-value* in [15]). In practice, it is typical to sum these two values, however, this framework lends itself well to implementing more sophisticated methods for generating a node's overal score.

B. Utilizing Perception Information

Modern perception systems are able to provide reliable information about the environment much farther from the vehicle than their predecessors from the mid-nineties, when arcs-based navigation was introduced. To stay up with this progress, navigators have increased the number and length of arcs, augmented them with paths of other shapes, e.g. clothoids [10]. There have been attempts to implement search-based selection of arcs in [12] in the limited region in front of the vehicle. We suggest that unlimited nonholonomic planning improves considerably on the previous attempts to explore all perceived environment more fully. As we have described, the unconstrained search that is the core of this planner is able to find plans all the way to the perception horizon, and sample the state space densely at any practical resolution. Moreover, if the vehicle had to stop in especially complex terrain (e.g. a cul-de-sac), the planner would naturally consider backward arcs and generate a steering maneuver that will guide the vehicle out of the impasse while satisfying its mobility constraints, all in minimal time.

To illustrate this discussion, we have chosen several examples of navigation in natural terrain that are problematic for arcs-based navigator and that suggest the improvements we propose. Figure 2a shows a large region of obstacles collinear with the vehicle and the goal. D* cost field is therefore symmetric, and thus two very different D* paths appear to have effectively the same cost. Any fluctuation of the vehicle pose toward one or the other will cause oscillation between D* paths and the corresponding choice of the arc. In the worst case, the vehicle will not be able to recover and eventually will run into the obstacle. As we will see later, the plan generated using search will develop the true cost toward the goal that will incorporate vehicle kinematics constraints and by its virtue of exploring a larger area than the immediate neighborhood of the vehicle, it will be able to avoid such oscillation by finding absolute lowest cost given vehicle pose. Given such a plan, it is also natural to allow the vehicle to commit to executing it for at least the distance of reliable perception horizon – such a commitment will also disregard D* fluctuations for a short

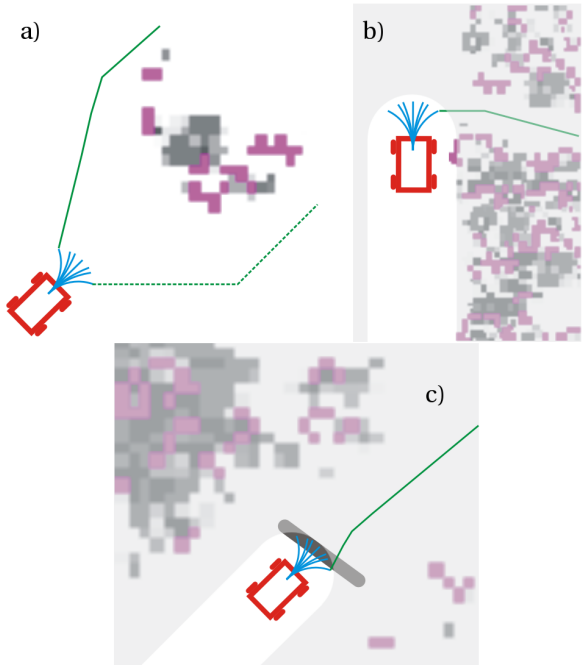

Fig. 2. Examples of typical challenges of arc-based navigators. White space represents flat terrain, gray regions are high cost and purple indicates obstacles. Figure a) represents a problem with oscillating D* path; b) shows that an attempt to execute a 90 degree turn suggested by D* can cause the vehicle to run into obstacles; c) illustrates how the lack of capacity to backtrack in simulation (as in planning via search) causes the vehicle to go over very high cost, whereas there is space to go around (if the robot state-space search, a maneuver to explore the area before commiting to this traversal would come out naturally).

while and thus enhance the robustness.

Another example occurs quite often in natural environments (Figure 2b). In this scenario of a narrow hallway that leads toward the goal. The arcs navigator will correctly lead the vehicle along the wall in search of the passage, however when one is discovered, it is typically much too late for the vehicle to make the turn. Another behavior would need to get involved to re-orient the vehicle into the passage. If, on the contrary, we imagine using search in the space of vehicle trajectories, then a necessary n-point maneuver to get into the hallway will be considered automatically (no need for special behaviors).

Figure 2c illustrates a situation where there is a high-cost, but not obstacle, area in front of the vehicle. It could, for example, be a fallen tree: the vehicle can go over it, but it is not easy and poses a risk of high-centering. The set of arcs is assumed to span the length of the perception horizon. The log is wide enough such that all arcs of the navigator go across it, thus all arcs have high cost and the arbiter has no chance of choosing an arc that would avoid the log. Unless special post-processing is implemented, such a navigator will generally enter all regions of high cost that are wider than the span of the arcs. It will correctly indicate that the vehicle must slow down, but nevertheless due to the "near-sightednes" of considering motion alternatives right in front, it will attempt to traverse such regions, which may be very costly and/or risky. Moreover, this does not appear to be the issue of limited perception horizon, because even if the robot had explored the area before, it would make the same mistake twice simply due to the nature of its navigator. In contrast, if the navigator had the capability to simulate further and backtrack, i.e. execute a search, then it could compare the cost of going over the high cost versus further exploring the area.

VI. Results

In this section we will focus on our current implementation of the navigator described herein and its comparison with other leading navigation systems. We discuss that while similar runtimes can be achieved by utilizing this motion planner as state-of-the-art navigators, the results are superior thanks to its quality of exploring much wider area around the vehicle, thereby exploiting all perception information that is avaialble. In this regard, the planner-enabled navigator outperforms its counterparts in the most complex and cluttered natural environments, i.e. where traversal success matters the most due to the risk of failure.

A. Runtime and Continuous Replanning

Typical navigation algorithm that allow continuous robot motion feature the ability to replan continuously. Arcs-based navigators have efficient arc selection algortihms that allow arc evaluation and arbitration to be executed continuously. In particular, [7] quotes the ability to perform perception-plannning cycles several times a second. In our implementation of the motion planner described here, we have achieved successful replanning rates of about 3 Hz, and so clearly continuous replanning is possible. Thanks to latest advances, nonholonomic motion planners approach the runtimes of simple predictive controllers.

We have also experimented with and obtained somewhat more efficient tranversal results by allowing the planner to run more slowly, but achieve lower bound on sub-optimality ϵ. In this case, the planner could be allowed to run for about a second. In order to support this longer runtime within the navigation scenario, we have utilized the approach of [12], where the search is started from the projected state of the robot given its current state. Further evaluation of this mode of operation will be the subject of the future work.

B. Simulation Results on Real Data

We have tested an implementation of the planner-enabled navigator presented herein in simulation in a variety of simulated and real-world environments. In easy to medium terrain, the performance of this navigator was similar to that of arcs-based navigators. This is likely due to a great deal of similarity between these algorithms. However, in difficult terrain, the planner-based navigator yielded more robust and efficient traversals. Due to the nature of the planner used, the traversal is not guaranteed to be optimal, but this navigator never failed to find a path to goal when it existed, which was a significant advantage over the tested arcs-based navigator.

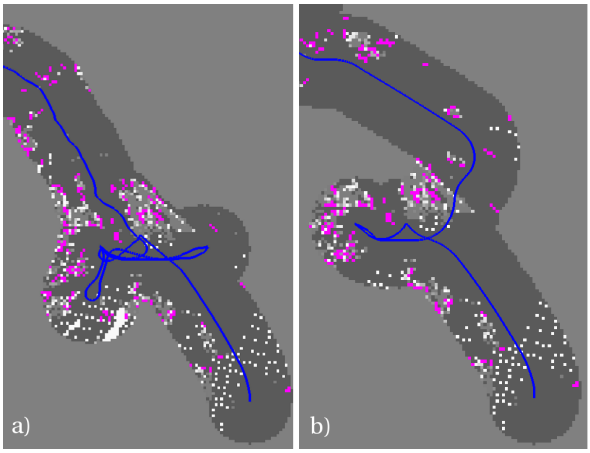

Fig. 3. Simulation of implementations of an arcs-based navigator (a) and the navigator utilizing motion planning via lattice search (b). This experiment features real perception data collected during field tests with the LAGR robots at JPL (courtesy of Andrew Howard).

In Figure 3 we show an example of a fairly difficult narrow passage in an obstacle-ridden environment (obtained from real data from field tests with the LAGR robots at JPL). Note that the LAGR robots feature differential-drive mobility, and so they can execute high-curvature turns and turn in place. Both navigators experienced difficulty traversing this terrain (middle of figures), while the planner-enabled navigator solved this problem with only a couple of point-turns. Thanks to the capability to perform unlimited motion planning "natively", the proposed navigator offers significant gains in robustness of traversing unknown rough terrain. This advantage is crucial for reliable autonomy and increased utility of off-road mobile robots.

VII. Conclusion

We have presented a robust navigation algorithm for mobile robots traversing unknown natural terrain. The goal of this work was to build on previous field experience as well as the latest in research in robot navigation and nonholonomic motion planning. Thanks to the increased computation availability and latest advances in motion planning research it becomes possible to perform full state-space search within the navigation framework. We provide special run-time performance and response characteristics to ensure anytime qualities of the motion planner, such that the planning can be done under tight timing constraints. Future work includes validation of this technology on real vehicles and further improving planning efficiency to increase replan frequency.

References

[1] Chakrabarti, P., Ghosh, S. and DeSarkar, S. Admissibility of AO* when heuristics overestimate. *Artificial Intelligence*, 34:97-113, 1988.

[2] Dean, T. and Boddy, M. An analysis of time-dependent planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1998.

[3] Ferguson, D., Stentz, A. Field D*: an interpolation-based path planner and replanner. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, October, 2005.

[4] Frazzoli, E., Dahleh, M.A. and Feron, E. Real-time motion planning for agile autonomous vehicles. In *Proceedings of the American Control Conference*, 2001.

[5] Goldberg, S., Maimone, M. and Matthies, L. Stereo vision and rover navigation software for planetary exploration. In *Proceedings of the IEEE Aerospace Conference*, vol. 5, 2002.

[6] Howard, T. and Kelly, A. Trajectory generation on rough terrain considering actuator dynamics. In *Proceedings of the International Conference on Field and Service Robotics*, August, 2005.

[7] Kelly, A., Amidi, O., Happold, M., Herman, H., Pilarski, T., Rander, P., Stentz, A., Vallidis, N., Warner, R. Toward reliable off-road autonomous vehicles operating in challenging environments. In *Proceedings of the International Symposium on Experimental Robotics*, 2004.

[8] Kelly, A. and Nagy, B. Reactive nonholonomic trajectory generation via parametric optimal control. *International Journal of Robotics Research*, 22:583-601, 2003.

[9] Kelly, A. and Stentz, A. Rough terrain autonomous mobility - part 1: a theoretical analysis of requirements. *Autonomous Robots*, 5:129-161, 1998.

[10] Kelly, A. and Stentz, A. Rough terrain autonomous mobility - part 2: an active vision, predictive control approach. *Autonomous Robots*, 5:163-198, 1998.

[11] Korf, R. Linear-space best-first search. *Artificial Intelligence*, 62:41-78, 1993.

[12] Lacaze, A., Moscovitz, Y., DeClaris, N. and Murphy, K. Path planning for autonomous vehicles driving over rough terrain. In the *Proceedings of the IEEE ISIC/CIRA/ISAS Joint Conference*, September, 1998.

[13] Latombe, J.-C. *Robot motion planning*. Kluwer, Boston, 1991.

[14] Likhachev, M., Ferguson, D., Gordon, G., Stentz, A. and Thrun, S. Anytime Dynamic A*: an anytime replanning algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, June, 2005.

[15] Likhachev, M., Gordon, G. and Thrun, S. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, MIT Press, 2003.

[16] Pancanti, S., Pallottino, L., Salvadorini, D. and Bicchi, A. Motion planning through symbols and lattices. In *Proceedings of the International Conference on Robotics and Automation*, 2004.

[17] Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[18] Pivtoraiko, M. and Kelly, A. Constrained motion planning in discrete state spaces. In *Proceedings of the International Conference on Field and Service Robotics*, August, 2005.

[19] Pivtoraiko, M. and Kelly, A. Generating near-minimal spanning control sets for contstrained motion planning in discrete state spaces. In *Proceedings of the International Conference on Intelligent Robots and Systems*, August, 2005.

[20] Simmons, R., Henriksen, L., Chrisman, L. and Whelan, G. Obstacle avoidance and safeguarding for a lunar rover. In *AIAA Forum on Advanced Developments in Space Robotics*, Madison, WI, August 1996.

[21] Zilberstein, S. and Russell, S. Approximate reasoning using anytime algorithms. In *Imprecise and Approximate Computation.* Kluwer Academic Publishers, 1995.