Model-Predictive Motion Planning

Several Key Developments for Autonomous Mobile Robots

By Thomas M. Howard, Mihail Pivtoraiko, Ross A. Knepper, and Alonzo Kelly

A necessary attribute of a mobile robot planning algorithm is the ability to accurately predict the consequences of robot actions to make informed decisions about where and how to drive. It is also important that such methods are efficient, as onboard computational resources are typically limited and fast planning rates are often required. In this article, we present several practical mobile robot motion planning algorithms for local and global search, developed with a common underlying trajectory generation framework for use in model-predictive control. These techniques all center on the idea of generating informed, feasible graphs at scales and resolutions that respect computational and temporal constraints of the application. Connectivity in these graphs is provided by a trajectory generator that searches in a parameterized space of robot inputs subject to an arbitrary predictive motion model. Local search graphs connect the currently observed state-to-states at or near the planning or perception horizon. Global search graphs repeatedly expand a precomputed trajectory library in a uniformly distributed state lattice to form a recombinant search space that respects differential constraints. In this article, we discuss the trajectory generation algorithm, methods for online or offline calibration of predictive motion models, sampling strategies for local search graphs that exploit global guidance and environmental information for real-time obstacle avoidance and navigation, and methods for efficient design of global search graphs with attention to optimality, feasibility, and computational complexity of heuristic search. The model-invariant nature of our approach to local and global motions planning has enabled a rapid and successful application of these techniques to a variety of platforms. Throughout the article, we also review experiments performed on planetary rovers, field robots, mobile manipulators, and autonomous automobiles and discuss future directions of the article.

Digital Object Identifier 10.1109/MRA.2013.2294914

Date of publication: 12 February 2014

Mobile Robot Motion Planning

The development and application of kinodynamic motion planning algorithms has been a key development in the last decade of autonomous mobile robots. Robots that can effectively model the consequences of their own actions can navigate more safely and efficiently in challenging environments. This is particularly important as these systems move outdoors into unstructured domains where the mapping between control inputs and robot motion can be nontrivial. Examples of such disturbances include terrain shape for planetary rovers and wheel slip for autonomous automobiles

1070-9932/14/$31.00©2014IEEE

and field robots. As it is difficult to generate trajectories by inverting the nonlinear, coupled equations of robot motion with limited computational resources, efficient sampling techniques must be developed to achieve the real-time performance in practical applications. This challenge drives applications away from naturally recombinant structures and toward sampling-based techniques that search in the space of sequential robot inputs. We assert that there is a need for a universal planning, navigation, and control framework that applies broadly across a wide range of platforms.

This article serves as an accessible summary of a decade of work on a number of programs that relate to the use of high-fidelity predictive models in mobile robot planning, navigation, and control. Herein, we review our model-based approach to trajectory generation, model calibration, and graph design with applications for local and global motions planning. We achieve generality by hiding the implementation of the predictive motion model from the optimizer, enabling rapid and successful deployment of such techniques on a diverse set of mobile robot systems (Figure 1).

Trajectory Generation

A trajectory generator must be able to determine the action $(\boldsymbol{u}(\boldsymbol{x}, t))$ that satisfies a set of state constraints $(\boldsymbol{x}_C(t))$ or minimize a cost function $(J(\boldsymbol{x}, t))$ or both subject to initial state constraints $(\boldsymbol{x}(t_0))$ and the predictive motion model $\dot{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{u}, t)$. Many techniques have been proposed, developed, and applied to solve this constrained optimization problem for the mobile robots. Our approach, detailed in [1], transforms the general problem of optimal control to parametric optimal control by parameterizing the space of inputs (1). This technique reduces searching in the space of all possible actions to a more computationally effective representation where the shape of inputs is controlled by a small number of degrees of freedom

$$\boldsymbol{u}(\boldsymbol{x}, t) \rightarrow \boldsymbol{u}(\boldsymbol{p}, \boldsymbol{x}, t). \tag{1}$$

Given an initial guess of action parameters $\boldsymbol{p}_0$, the trajectory generation problem becomes one of finding a correction $\Delta\boldsymbol{p}$ that satisfies the following expression:

$$\boldsymbol{x}_C(t_1) = \int_{t_0}^{t_1} \dot{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{p}_0 + \Delta\boldsymbol{p}, \boldsymbol{x}, t), t)\, dt. \tag{2}$$

Solving this expression directly is impractical, if not impossible, for the mobile robots operating in realistic three-dimensional (3-D) terrain with nontrivial actions. For mobile robots operating on planar surfaces that are even, (2) contains trigonometric expressions that render it nonintegrable for nonconstant linear and angular velocities. For physical systems that navigate on rough (two-and-a-half or 3-D) terrain, it becomes nonintegrable in any case. Our approach alternatively generates numerical estimates of the Jacobian and Hessian to iteratively refine the initial guess until the boundary state constraints are satisfied, the minimum-cost solution is found, or both. The partial derivatives in these matrices are found using forward simulations of the predictive motion model with small perturbations of individual action parameters. When it is not necessary to minimize a cost function in addition to satisfying terminal boundary state constraints, the expression in (2) can be linearized and inverted to estimate a set of parameter corrections that satisfies the initial state, predictive motion model, and terminal boundary state constraints

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i - \left[\frac{\partial \Delta\boldsymbol{x}(\boldsymbol{p}_i)}{\partial \boldsymbol{p}_i}\right]^{-1} \Delta\boldsymbol{x}(\boldsymbol{p}_i) \qquad i \geq 0. \tag{3}$$

One of the challenges of applying such techniques for mobile robot trajectory planning involves how the parameter space is reduced to search efficiently and avoid local optima. Some of the initial work on this topic applied polynomial functions of curvature parameterized by signed distance to express the space of candidate vehicle motions [2]. These expressions were well suited to indoor mobile robot applications because, for simple kinematic predictive motion models, the shape of the

A formulation that calibrates the predictive motion model can be used either online or offline, by observing its integrated effect over short time intervals, and comparing that result to direct measurements.

Figure 1. The experimental platforms used during the evolution of our hierarchical trajectory planning approach: (a) Rocky 8, (b) Crusher, (c) LAGR, and (d) Boss.

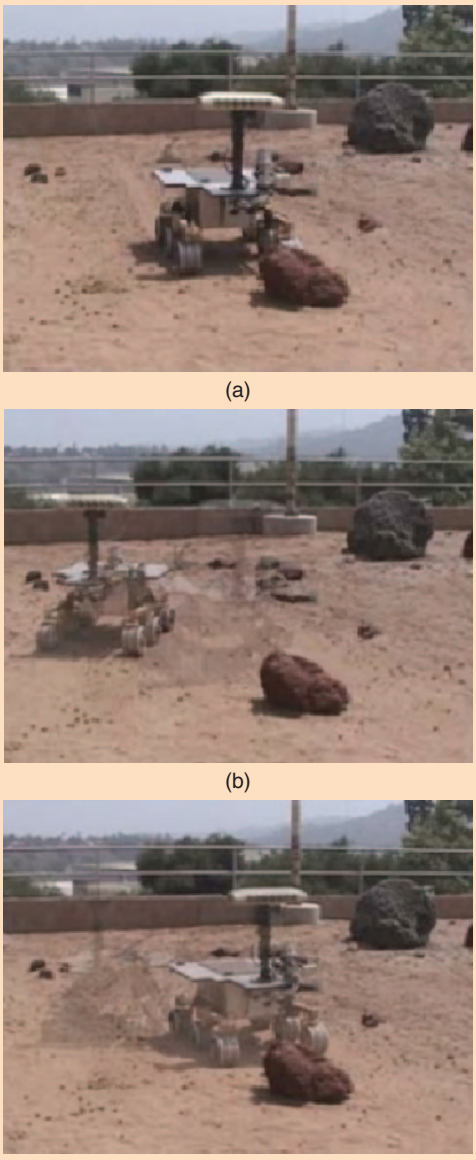


Figure 2. Planetary rover experiments of model-predictive trajectory generation: (a) normal mobility, (b) impaired mobility, and (c) impaired mobility with predictive compensation.

trajectory is not influenced by the speed at which it is driven. For more sophisticated predictive motion models that account for dynamic effects, scaled solutions provide excellent initial guesses that can be corrected using the discussed techniques. One weakness of this parameterization for trajectory generation and optimization involved the varying sensitivity of variables in these expressions. This sensitivity made it difficult to accurately estimate the derivatives numerically because of the scale at which it is necessary to perturb action parameters may vary significantly. An alternate parameterization based on curvature splines provides better convergence performance, as the sensitivity of predictive motion model simulation to small changes in action parameters is more uniform. Variations of this technique for unconstrained and constrained optimization trajectory planning, along with more detail about the constrained trajectory planning algorithm and a review of the comparative literature is described in [1].

An example of this technique applied to motion planning in rough terrain for a plantetary rover with a rocker-bogie chassis is illustrated in Figure 2. Here, we solve for the parameters of an action that satisfies a terminal boundary state constraint that requires the robot to be positioned 4 m forward and 1 m to the left of the initial state and a predictive motion model that simulates the response of the rocker-bogie chassis with a two-and-a-half-dimensional terrain. Assuming a first-order spline function of curvature, a constant velocity function parameterized by distance traveled, and an initial guess of parameters ($p_0 = [\kappa_1, s_f] = [0.0\ \text{rad}, 3.0\ \text{m}]$), a solution with subcentimeter accuracy is found in just three iterations. Since the partial derivatives of constraint error with respect to parameterized freedom values are determined numerically, the algorithm is able to invert a sophisticated model of robot-terrain interaction that simulates the response of the mobility system with the terrain.

To demonstrate this approach in an applicable domain, we applied this technique on a planetary rover operating in the Jet Propulsion Laboratory (JPL) Mars Yard. We created a disturbance in the robot's motion model by disabling the rear right drive motor in software. This mirrors the experience on Spirit, one of the two planetary rovers from the Mars Exploration Robots mission, where one of the drive motors was dragged to conserve motor lifetime [3]. There are several ways to compensate for this disturbance, with the most common being feedback control. In applications where a global-positioning system is unavailable, the fusion of odometry and visual feedback can be used to better estimate the current state. Although a recent article has pushed toward more efficient vision-based pose estimation and navigation [4], it still can be computationally prohibitive to estimate state at rates required for feedback control. To address this problem, we simply adapt the predictive motion model used by our trajectory generator to reason about the disturbance. The solution found by the trajectory planner correctly computed actions that drive the vehicle harder to the left, balancing the influence of the disabled drive wheel. Images of this experiment are shown in Figure 3, which illustrate tens of centimeters of error without predictive compensation, and only a few centimeters of error with predictive compensation. More details of this experiment can be found in [5].

The utility of a capacity to invert models of mobility and utilize this form of predictive error compensation depends strongly on the accuracy of the models that are being inverted. Luckily, the same mechanism of parameterization can be used to calibrate the models used, based on observations generated while the system operates. Furthermore, the processing requirements for model identification are trivial compared with the predictions being computed in most planners already. A formulation that calibrates the predictive

motion model can be used either online or offline, by observing its integrated effect over short time intervals, and comparing that result to direct measurements [6].

The reader may wonder why a model is needed if measurements are available. There are two reasons, both derived from the fact that a predictive model is desired that predicts motion from commands, rather than estimating it from measurements. When calibrating online, the measurements are not available until after the motion has been executed. The process of continuously comparing predictions with measurements refines and adapts the model. Conversely, when calibrating offline, the measurements can be generated based on high-performance ground-truth sensing, perhaps based on infrastructure, which would not be available during normal operation.

While many alternatives exist, we have used a formulation that associates all prediction error with presumed disturbance inputs. We assume that the disturbances $\boldsymbol{\delta u}(\boldsymbol{p},\boldsymbol{x},\boldsymbol{u})$ depend on some parameters and the state and inputs u. For example, the commands may be linear V and angular velocity ω. The attitude of the vehicle (ϕ,θ), or equivalently the gravity vector, is also considered to be an aspect of the state. Linear and quadratic terms are included in the model. In the following example expression, the $V\omega$ term captures the effect of lateral acceleration:

$$\boldsymbol{\delta u}(\boldsymbol{p},\boldsymbol{x},\boldsymbol{u}) = p_1 V + p_2\omega + p_3 V\omega + p_4 \sin(\phi) + \cdots. \quad (4)$$

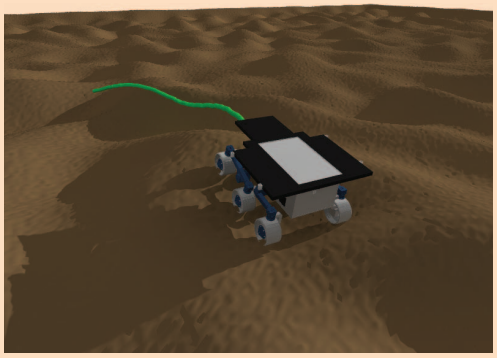
The disturbances can be visualized as wheel slip, but because all prediction error is being calibrated, this means that other sources of error are being converted to *equivalent* wheel slip. All velocities in the state are expressed in body coordinates so that they are constant under steady state turning conditions. In the systematic model, the task is to predict the system state at future time t_1

$$\boldsymbol{x}(t_1) = \int_{t_0}^{t_1} \dot{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{u} + \boldsymbol{\delta u}(\boldsymbol{p},\boldsymbol{x},\boldsymbol{u}))dt. \quad (5)$$

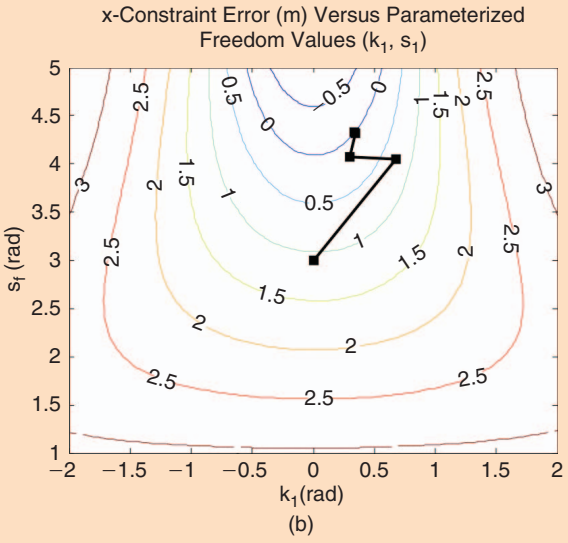
The Jacobian of this state prediction integral is, once again, computed numerically. For offline use, multiple observations can be used to generate an overdetermined system of equations to be solved for the disturbance parameters. A more elegant method would be formulating an identification Kalman filter, which runs continuously while the system operates.

Local Search Graphs

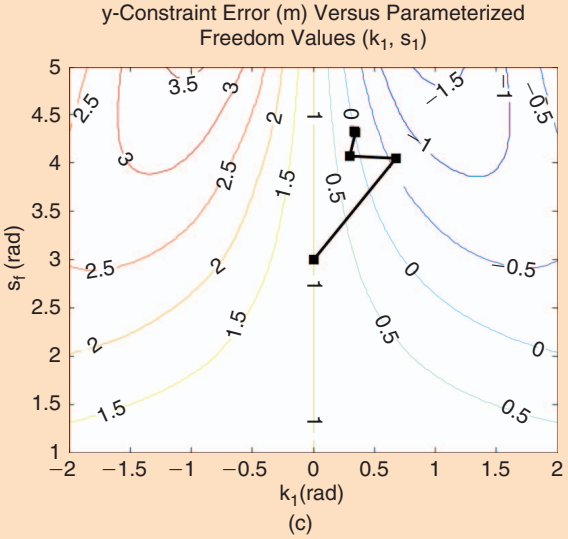
Local motion planning and obstacle avoidance is the problem of search in the space of horizon-limited actions to select a motion that keeps a mobile robot safe and moving in a way that satisfies some objectives. The planning horizon can be described using distance or time, but it should extend to or beyond the stopping distance of the platform in the local environment. The predictive motion model used for the local motion planner must also reasonably approximate the vehicle response to a set of actions because feedback control can, in certain situations, execute maneuvers



(a)



(b)



(c)

Figure 3. An example of model-predictive trajectory generation applied to motion planning in rough terrain for a planetary rover with a rocker-bogie chassis: (a) shows the generated trajectory that satisfies the terminal state, initial state, and motion model constraints, and (b) and (c) show contour plots of the constraint error with respect to parameterized freedom values and an overlay of the history of parameter corrections by the trajectory generation algorithm.

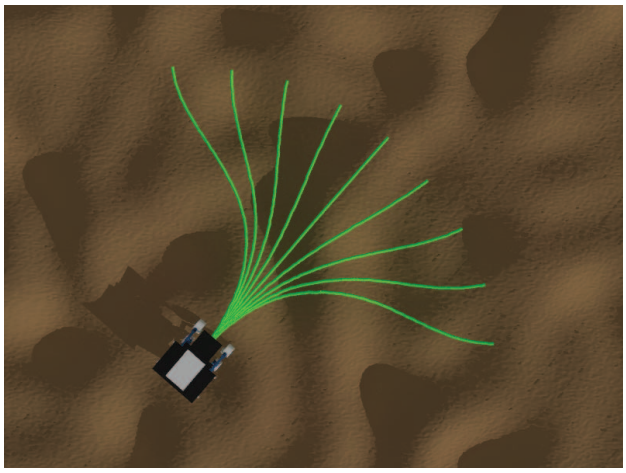


Figure 4. A local motion planning search space is generated by sampling with uniform distribution of terminal poses.

inadequately. It is also necessary that the local motion planning and obstacle avoidance algorithms react quickly to real-time perception and world model information. Given limited computational resources and temporal constraints, it is essential that the local search graph be well separated, feasible, and efficient.

Standard methods for generating local search graphs are based on sampling in the action space. For these methods, trees of varying length are generated by deliberative [7] or randomized sampling [8] in the space of vehicle inputs. One significant advantage of this approach includes the inherent feasibility of sampled motions. The main disadvantage is that it can be difficult or computationally expensive to generate a well-separated search space. This is because the state-space response of a sampled input can vary as a function of vehicle state. An alternative approach samples in the space of constraints, defining a set of boundary states that a robot must achieve. We use the aforementioned trajectory generation techniques to efficiently generate the actions in real time that satisfy constraints imposed from sampling and the robot's own environmental interaction [9]. The main advantage of this technique is the ability to generate a set of sophisticated maneuvers with only a small number of trajectory generation queries. Consider the example shown in Figure 4. Each of these trajectories satisfies initial posture (x, y, ψ, κ) and terminal pose (x, y, ψ) constraints using a curvature spline with four degrees of freedom $(\kappa_0, \kappa_1, \kappa_2, s_f)$. An input-space sampling technique would have to sample in each of these four dimensions to produce a set that contains these nine diverse maneuvers.

Simulation and field experiments demonstrated performance improvements over the input-space sampling techniques in comparative simulation and field tests.

Another advantage to this approach is that if global guidance is available or environmental constraints are known, the rules for distributing the boundary state constraints can be adjusted to increase the likelihood of generating a collision-free or desirable motion.We experimented with this technique on a field robot, an autonomous automobile, and a mobile manipulator.

Crusher, pictured in Figure 1(b), is a six-wheeled skid-steered field robot that can navigate autonomously in unstructured outdoor environments. When traveling quickly, the angular velocity constraints of the identified predictive motion can have a great influence on the state-space response of the action. We observed that trajectories uniformly sampled in the input space cluster together during turns, requiring denser sampling in the input space to increase the likelihood of finding a collision-free path in cluttered terrains. By sampling in the space of terminal position and orientation constraints, we were able to generate expressive local search graphs that were far less sensitive to the influence of the initial state. Figure 5(a)–(c) shows three local search graphs with varying initial angular velocities. Simulation and field experiments demonstrated performance improvements over the input-space sampling techniques in comparative simulation and field tests. Details of the experiments involving our approach to local search graphs on Crusher can be found in [9].

Boss, pictured in Figure 1(d), is an autonomous automobile that won the Defense Advanced Research Projects Agency (DARPA) Urban Challenge [10]. Local search graph construction techniques are particularly important on such platforms, as they must operate safely at high speeds and avoid collisions with people, vehicles, and structures in a constrained environment. One important characteristic of this platform's mobility system was the conservative bounds on the allowable rates of curvature imposed by the commercial off-the-shelf drive by a wire system. This meant that for the vehicle to avoid an obstacle at high speed, it must begin turning the wheel well before reaching the impediment. We applied the constraint sampling technique and knowledge from the perceived world model to generate trajectories that kept the vehicle in-lane and satisfied the constraints of our observed predictive motion model (Figure 6). To increase the diversity of candidate maneuvers in the local search graph, we added constraint on the initial curvature command to produce both sharp (discontinuous) and smooth (continuous) curvature commands for the same terminal boundary state constraint. The constraint sampling technique also simplified the problem of lane changing by sample constraints that produce a pose in the center and direction of the neighboring lane. More details about the application of our approach to a local search graph for in-lane navigation on Boss can be found in [11].

Applications of these techniques for generating feasible and well-separated local search graphs are not limited to

terrestrial vehicles operating in outdoor environments. We can apply these methods to mobile manipulators by augmenting the search space with configuration constraints of the manipulator. Figure 7 illustrates slices of a local search graph generated by sampling constraints that included the terminal position of the end-effector and orientation of the base with respect to the initial pose. We applied this technique for intelligent teleoperation of a Learning Applied to Ground Robots (LAGR) platform mobile robot [12] outfitted with a five degrees-of-freedom manipulator to autonomously approach objects of interest. More details on experiments involving the application of our approach to local search graphs on mobile manipulators can be found in [13].

For each of these robots, we constructed local search graphs by sampling in the space of boundary state constraints. The rules for how boundary state constraints are distributed in the local search graph are dependent on the characteristics of the robot, application, and environment. This flexibility, coupled with the capacity to efficiently generate trajectories subject to an arbitrary predictive motion model, provides us with a simple set of tools for generating local search graphs that are well separated, feasible, and efficient to compute.

Global Search Graphs

Global search graphs represent chains of possible decisions that are searched to find a minimum-cost path from the initial state to some set of goal states. These approximations of the space of all robot motions typically exhibit longer sequences of robot decisions, lower fidelity of robot motions, and coarser resolution of robot state than their local search graph counterpart. Search in global search graphs typically involves search implemented with data structures with *divergent* (acyclic) or *recombinant* (cyclic) connectivity. Common examples of divergent structures are trees, which can be generated using the previously described input-space sampling methods. A well-studied implementation of divergent motion planning search spaces is the rapidly exploring randomized tree [8], where heuristics are used to bias sampling toward unexplored regions. Grids are the most common type of recombinant *graph* search space. Vertices corresponding to state values are regularly sampled throughout the state space, and edges express the cost and existence of the motions required to transition between adjacent vertices.

Our approach to global search graph construction uses a special type of trajectory library composed of motions that satisfy differential constraints and conforms to a regularly distributed set of boundary states [14]. The position dimensions are sampled as a uniform grid (rectangular, hexagonal, or other repeating unit). The choice of grid resolution significantly depends on specifics of the system: it is chosen to be commensurate with the trajectory following accuracy of the robot controller and the feature size of the robot's perception system (e.g., for occupancy grid approaches, the resolution of this grid). Intuitively, search fixed to the grid resolution could

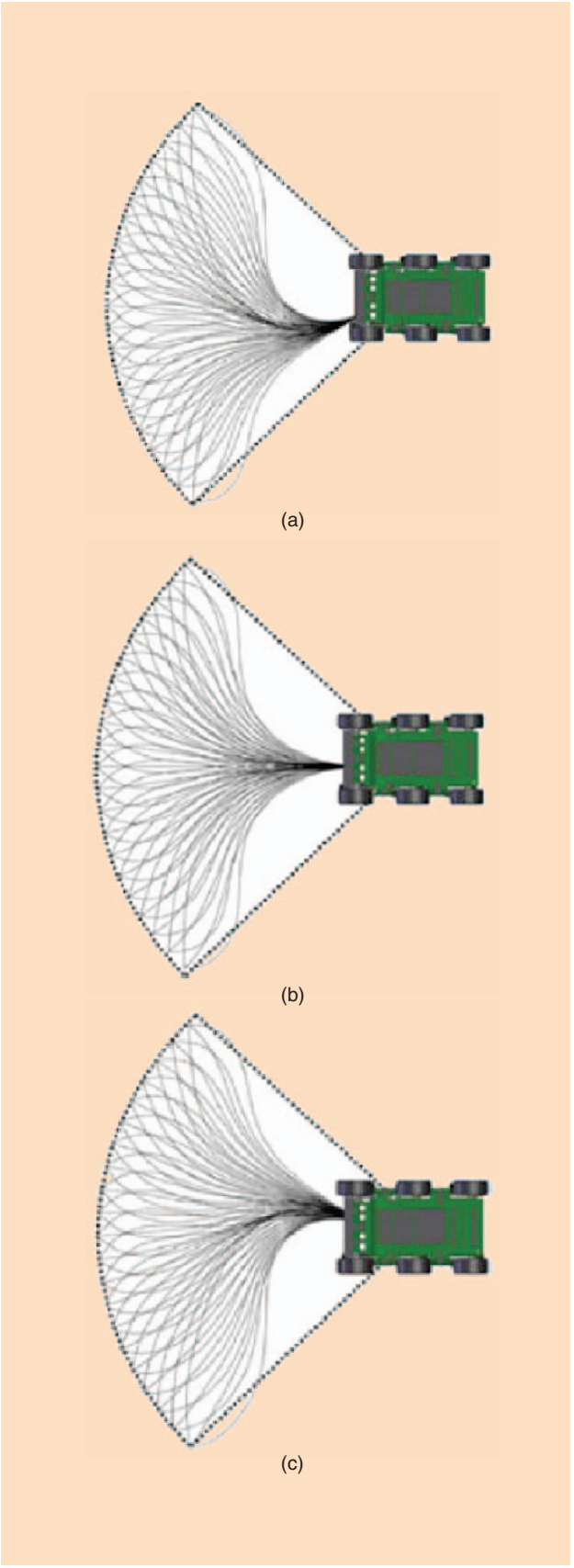


Figure 5. Local search graphs with varying initial angular velocities subject to uniformly sampled terminal boundary constraints: (a) turning left, (b) driving straight, and (c) turning right.

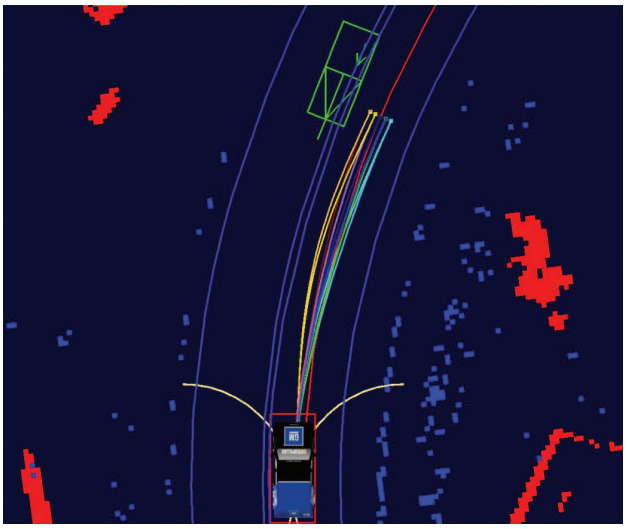
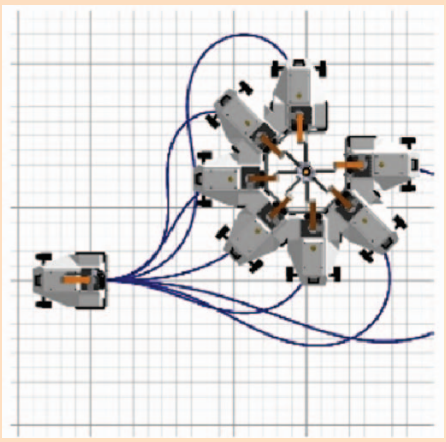
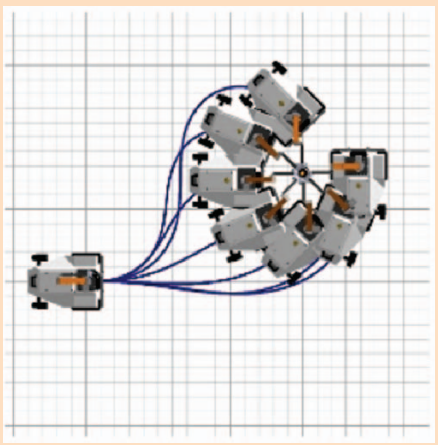


Figure 6. Local search graphs are generated with sampled constraints that consider features of the environment applied to an autonomous automobile. Note that lane constraints are respected by all trajectories at a distance of approximately three vehicle lengths.



(a)



(b)

Figure 7. Local search graphs are generated with sampled constraints that apply to both the mobility system and the manipulator. (a) Base aligned with object. (b) Base offset from object.

be too expensive to compute (if the resolution is too high) or not expressive enough (if the resolution is too low).

Assuming lattice state discretization outlined earlier, motion primitives are defined here to be the controls that connect roadmap vertices (states) and that are feasible motions. Furthermore, enforcing state continuity requires that the state dimension vector be augmented with appropriate dimensionality. A simple example of such a search space is the state lattice for a Reeds–Shepp car [15] in Figure 8. This system has carlike kinematics, with a restriction that its steering angle can have only three values, $[-\phi_{\max}, 0, \phi_{\max}]^T$, where $\phi_{\max}$ is the maximum steering angle of the robot. A slightly more involved example of a state lattice for a carlike robot without such a restriction is shown in Figure 9.

The process of constructing the state lattice assumes a capability to determine a control to the system that steers it from one state to another. Exact solvers exist for simple systems [15], but a number of other approximate solutions techniques, such as the trajectory generator presented earlier in this article, could also be applied. Implementation of a state lattice is a two-step process. First, we design a control set that defines the local connectivity of a state lattice by generating trajectories to neighboring states in a regularly sampled state-space. Second, we repeatedly expand the control set within a search process to find a minimum-cost motion that satisfies differential constraints.

It is widely accepted that one of the most computationally intensive procedures in planning is collision detection and estimation of motion cost. To compute this cost accurately, it is necessary to simulate the behavior of the vehicle, subject to the corresponding control in its environment. The cost estimation is achieved by convolving the vehicle frame along the workspace projection (path) of the control trajectory in question. Since we precompute motion alternatives, their paths (workspace projections) can be cached as well, perhaps in the form of trajectory swaths, the set of cost map cells $C_s \subset \mathbb{N}^2$ that are occupied by the robot footprint during motion (gray cells in Figure 10). Hence, instead of the costly vehicle simulation, edge cost computation can be reduced to accumulating the cost over an array of map cells, resulting in potential orders of magnitude speed-up in overall planning.

Well-informed search heuristics have the potential to increase the efficiency of the search substantially [16]. Developing good heuristics for planning with differential constraints is a challenging problem. Among the simplest options for a heuristic estimate in the given context is the Euclidean distance metric. Weighted Euclidean distance, where dimensions are scaled differently, and Mahalanobis distance can also be applied. Such metrics are computationally efficient and can be defined to satisfy the *admissibility* requirement [16]. However, in many cases, such approaches vastly underestimate the true path length in the lattice, resulting in inefficient search.

Better informed heuristics can be designed by using the vehicle controller. Since this entity is typically endowed with the system model, it has the potential of estimating the heuristic

cost of a motion more accurately. However, even though a heuristic based on Reeds–Shepp distance [15] is a much better informed one for a carlike robot than the Euclidean distance, it still can suffer from significant inaccuracies.

Similar to trajectory swaths in the previous section, state lattice structure enables a straight-forward precomputation of the free-space costs of motions, stored in a look-up table, leading to a well-informed free-space heuristic in terms of actual cost of feasible motions. Such a heuristic look-up table (HLUT) can be set up as a multidimensional array, where elements are scalars that represent the cost of steering the system from the origin to a collection of states in free space [17], [18]. An inherent limitation of this approach, however, is the memory requirement for storage. Even though the availability of affordable computer memory is continuously increasing, it is interesting to seek methods to alleviate this requirement. One approach is to utilize an *approximate* HLUT that has more sparse sampling than the planner search space.

For mobile robots operating in cluttered environments, it is frequently beneficial to utilize a high density of representation in the immediate vicinity of the robot (perhaps within its sensor range) and to reduce the density in the areas that are either less known or less relevant for the planning problem (Figure 11). A lower density of representation is designed to increase search speed, whereas higher density provides better quality solutions. Since grids have traditionally been utilized in replanning, the notion of varying the quality of problem representation has often been identified with varying the resolution of the grid. However, here we emphasize the discretization of a continuum of possible motions. Hence, we refer to managing the density of the state lattice representation as *graduated density.*

In designing the connectivity of regions of different densities, care must be taken to ensure that all densities consist of motions that are feasible with respect to the robot's predictive motion model. If this rule is violated, mission failures become possible because of limit cycling at the border of density regions due to obstacles. To avoid such difficulties, it is sufficient to ascertain that all levels of density include feasible motions. For example, the connectivity of low-density regions could be a strict subset of the high-density regions.

To verify that the runtime performance of the proposed precomputed search spaces is acceptable for the intended application in field mobile robots, a comparison with several basic grid search planners was undertaken. It is challenging to compare these planning approaches due to their fundamental differences, nevertheless their similarity in runtime, as shown below, suggests that the proposed strategy may be a beneficial one.

The experimental comparison is set up with four different control sets, in each case using a heuristic function that returns the exact distance to the goal. The basic state lattice is used along with a large HLUT. Three grid control sets are tested, in which each state is connected to its four, eight, and 16 nearest neighbors, using a perfect heuristic for each level of connectivity. Results in a simulated world with point obstacles

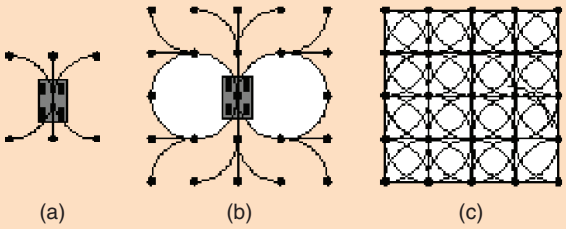


Figure 8. The Reeds–Shepp car can move forward and backward, and it can drive straight or turn left or right at a fixed curvature. A rudimentary control set shown in the figure is derived from these basic controls by choosing their length such that their terminal points coincide with a regular lattice sampling in state space. (a) A simple set of six motion primitives for a carlike robot that comply with the state lattice structure. (b) A subset of a lattice graph composed by duplicating motions from (a). (c) An excerpt of the fully connected lattice graph obtained via further duplication of the elementary set of motion primitives.

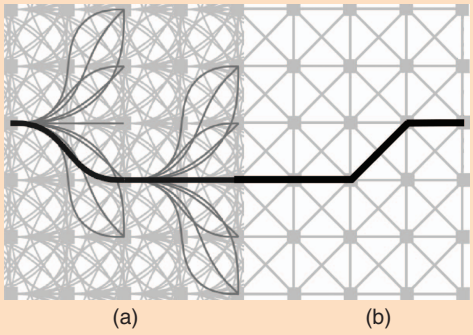


Figure 9. A simplified representation of two state lattices of different densities, seamlessly merged due to lattice structure: (a) a 3-D roadmap that includes 2-D position (x, y) and heading is allocated in the vicinity of the vehicle, and (b) a lower-dimensional one (position only) is further away.

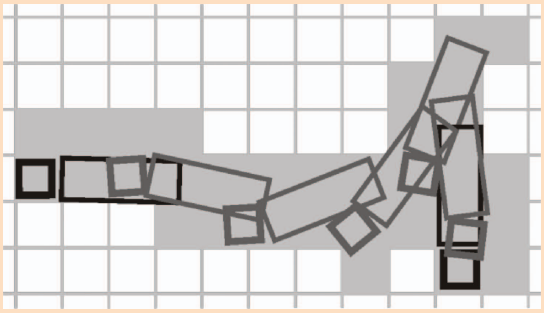


Figure 10. A tractor-trailer motion primitive reorients the robot by 90° in heading. The trajectory swath was computed using Runge–Kutta simulation and cached offline. During the online planning, the evaluation of the cost of this maneuver with respect to a cost map is reduced to a summation operation over a memory array, resulting in an approximately 100x speed-up over the simulation.

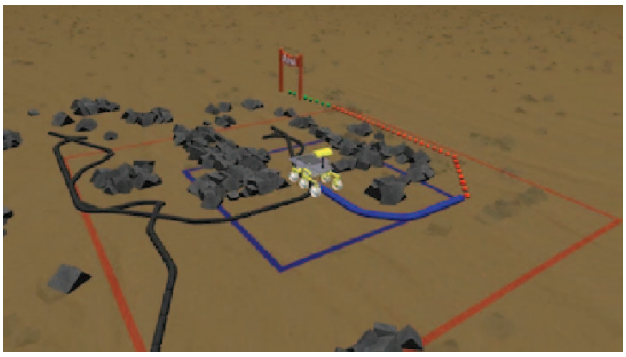


Figure 11. The graduated density lattice planner operating in a simulated environment using a planetary rover model is illustrated. The search graph inside the blue boundary uses maneuvers that are continuous in curvature, the area between the blue and red boundaries search uses an eight-connected grid, and the area outside the red boundary uses the Euclidean distance to the goal.

Figure 12. The FIDO rover navigated autonomously among previously unknown obstacles using the graduated density lattice planner. The approximate path the rover followed is shown in white.

(size of a single grid cell) obstacles with 5% density suggest that grid planners outperform the lattice in this obstacle field because they disregard differential constraints. The constrained lattice planner, however, satisfies the given differential constraints (heading and curvature continuity, maximum curvature of $1/8 = 0.125$ cells). In plain terms, a grid planner produces faster answers, but they are usually wrong. Despite the somewhat increased computation requirements, the lattice consistently remains approximately one order of magnitude slower than the grid planner on average, it returns results in fewer than 0.1 s for all classes of queries on commodity computing hardware, which is acceptable for a wide range of applications. See [14] for further details on this and similar comparative studies of motion planning using state lattices.

The graduated density motion planner was integrated with research prototype rovers at the JPL. It enabled rovers to navigate in rough rocky terrain set up in the JPL Mars Yard.

The graduated density motion planner was integrated with research prototype rovers at the JPL. It enabled rovers to navigate in rough rocky terrain set up in the JPL Mars Yard. Figure 12 shows the results of the field-integrated design and operations (FIDO) rover running the graduated density state lattice planner on-board to navigate autonomously amid dense rocks. In this experiment, the robot featured a single 1.6-GHz CPU and 512 MB of RAM, shared among all processes of the rover. The actual memory usage of the planner was less than 100 MB. The rover used a high-density region of the same size as above, 21×21 map cells, and a perception region (via stereo cameras) 41×41 map cells, both centered around the rover. Figure 12 shows the approximate path that the rover traveled.

The experiments were designed to verify that the motion planner described herein is capable of generating feasible robot trajectories with respect to the robot's differential mobility constraints. The demonstrated real-time execution on standard computing hardware suggests this approach may find applications in other similar mobile robot missions. This result is made possible by establishing a high-quality sampling in state space and utilizing the local planning approaches from earlier sections to generate a compatible sampling in the control space. We observe that the proposed structured representation provides a number of algorithmic and computational optimizations that enable fast global planning in difficult environments. More importantly, automatic generation of near-optimal control sets that form a recombinant motion planning search space is straightforward within this framework.

Conclusions and Future Work

A capacity to both calibrate and invert general models of mobility renders mobile robots far more informed about their own capacity to maneuver. It leads to robots that can precisely determine the inputs necessary to acquire a desired terminal state and allows feedback control to concentrate on correcting for disturbances that cannot be predicted. Such a high-performance trajectory generator leads to approaches to local planning that are intuitive, straightforward, and effective. Here, the essence of the problem becomes how to effectively sample boundary conditions that shape the search space. For planning on a global scale, high-performance trajectory networks that encode sophisticated maneuvers can be synthesized efficiently through the careful design of a feasible trajectory library.

We have shown the benefits of this framework for planning, navigation, and control on a number of diverse mobile robot systems. Our work centers on the idea of developing trajectory planners that satisfy generalized predictive motion

models at all layers of the navigation architecture. Generality of predictive motion models is important, as it enables fast application on a broad number of systems, allows tuning of the fidelity to match the computational resources of the platform, and provides a means to exploit information derived from self-calibration.

The application of our research has opened other important areas for investigation. Methods for automatic parameterization of inputs that take into account the difficulty of the terrain and density of obstacles will simplify the application of the trajectory generator presented in this article. Techniques that permit adaptation of state lattices with graduated density will improve the quality of planned trajectories while reducing computational complexity of heuristic search. Extension and application of these algorithms for systems that do not operate on a 3-D surface (e.g., unmanned aerial vehicles, autonomous underwater vehicles) also remains an area of inquiry.

Acknowledgments

Portions of this article were funded by NASA, DARPA, Ford Motor Company, General Motors, and the U.S. Army. Detailed acknowledgment of sponsorship is available in [1], [2], [5], [6], [9], [11], and [13]. Part of the work described in this article was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract to the National Aeronautics and Space Administration. We would like to acknowledge Colin Green, Dave Ferguson, Max Likhachev, Dean Anderson, Forrest Rogers-Marcovitz, Neal Seegmiller, Issa Nesnas, Mike McHenry, and Hari Nayar for their help with applications of these algorithms.

References

[1] T. Howard and A. Kelly, "Rough terrain trajectory generation for wheeled mobile robots," *Int. J. Robot. Res.*, vol. 26, no. 2, pp. 141–166, Feb. 2007.

[2] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 199–217, 2002.

[3] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *J. Field Robot.*, vol. 24, no. 3, pp. 169–186, 2007.

[4] T. Howard, A. Morfopolous, J. Morrison, C. Villalpando, L. Matthies, and M. McHenry, "Enabling continuous planetary rover traverse through FPGA stereo and visual odometry," in *Proc. IEEE Aerospace Conf.*, 2012, pp. 1–9.

[5] M. Pivtoraiko, T. Howard, I. Nesnas, and A. Kelly, "Field experiments in rover navigation via model-based trajectory generation and nonholonomic motion planning in state lattices," in *Proc. 9th Int. Symp. Artificial Intelligence, Robotics, Automation Space*, 2008.

[6] N. Seegmiller, F. Rogers-Marcovitz, and A. Kelly, "Identification of vehicle models using integrated equation error," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 912–931, 2013.

[7] M. Maimone, J. Biesiadecki, E. Tunstel, Y. Cheng, and C. Leger, "Surface navigation and mobility intelligence on the Mars Exploration Rovers," in *Intelligence for Space Robotics*. San Antonio, TX: TSI Press, Mar. 2006, ch. 3, pp. 45–69.

[8] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[9] T. Howard, C. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *J. Field Robot.*, vol. 25, nos. 6–7, pp. 325–345, 2008.

[10] M. Buehler, K. Iagnemma, and S. Singh, "Editorial," *J. Field Robot.*, vol. 25, no. 8, pp. 423–424, 2008.

[11] D. Ferguson, T. Howard, and M. Likhachev, "Motion planning in urban environments," *J. Field Robot.*, vol. 25, nos. 11–12, pp. 939–960, 2008.

[12] L. Jackel, E. Krotkov, M. Perschbacher, J. Pippine, and C. Sullivan, "The DARPA LAGR program: Goals, challenges, methodology, and phase I results," *J. Field Robot.*, vol. 23, nos. 11–12, pp. 945–973, 2006.

[13] D. Anderson, T. Howard, D. Apfelbaum, H. Herman, and A. Kelly, "Coordinated control and range imaging for mobile manipulation," in *Proc. 11th Int. Symp. Experimental Robot.*, Aug. 2008, pp. 547–556.

[14] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *J. Field Robot.*, vol. 26, no. 3, pp. 308–333, 2009.

[15] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, 1990.

[16] J. Pearl, *Heuristics*. Boston, MA: Addison-Wesley, 1984.

[17] M. Pivtoraiko and A. Kelly, "Constrained motion planning in discrete state spaces," in *Proc. 5th Int Conf. Field Service Robotics*, 2005, pp. 269–280.

[18] R. A. Knepper and A. Kelly, "High performance state lattice planning using heuristic look-up tables," in *Proc. IEEE/RSJ Int. Conf. 945973 Intelligent Robots Systems*, 2006, pp. 3375–3380.

Thomas M. Howard, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. E-mail: tmhoward@csail.mit.edu.

Mihail Pivtoraiko, GRASP Laboratory, University of Pennsylvania, Philadelphia, Pennsylvania, USA. E-mail: mihailp@seas.upenn.edu.

Ross A. Knepper, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. E-mail: rak@csail.mit.edu.

Alonzo Kelly, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. E-mail: alonzo@cmu.edu.