Generating Near Minimal Spanning Control Sets for Constrained Motion Planning in Discrete State Spaces

Mihail Pivtoraiko and Alonzo Kelly

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
email: mihail@cs.cmu.edu, alonzo@ri.cmu.edu

Abstract

We propose a principled method to create a search space for constrained motion planning, which efficiently encodes only feasible motion plans. The space of possible paths is encoded implicitly in the connections between states, but only feasible and only local connections are allowed. Furthermore, we propose a systematic method to generate a near-minimal set of spatially distinct motion alternatives. This set of motion primitives preserves the connectivity of the representation while eliminating redundancy -- leading to a very efficient structure for motion planning at the chosen resolution.

1 Introduction

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 introduce 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 built here in an inverse manner by using an inverse path generator. For a moment, let us imagine that an ideal inverse path generator exists that, given initial and final configurations in an obstacle-free C-space, generates a feasible path between them or reports that no path exists.

The lattice is constructed by discretizing the C-space into a hyperdimensional grid and attempting to connect the origin with every node of the grid using a feasible path, an edge. The lattice in general is also assumed to contain all feasible paths, up to a given resolution, which implies that if it is possible for a vehicle to travel from one node to another node, then the lattice contains a sequence of paths to perform this maneuver. Hence, it is possible to conclude that this formulation allows resolution complete planning queries.

Like a grid, the state lattice converts the problem of planning in a continuous function space into one of generating a sequence of decisions chosen from distinct alternatives. Unlike a grid, the state lattice encodes no default assumptions about how states are connected. Conversely, it is typical to assume that adjacent cells in a grid are connected in a 4-connected or 8-connected arrangement.

Such default connectivity fails to capture nonholonomic constraints, leaving them to be considered heuristically in the optimization process during planning, or as an afterthought in plan post-processing. The lattice, however, has an important property that each connection represents a feasible path. A connectivity scheme that intrinsically represents mobility constraints leads to superior motion planning results because no time is wasted either generating, evaluating, or fixing infeasible plans.

As a time-unlimited reachability graph, the state lattice includes every node in the grid. We can visualize this by imagining growing the state lattice from the origin; at some time enough paths will be contained in the lattice to perform a parallel parking maneuver, essentially allowing the robot to move sideways. Thus, in order to encode nonholonomic constraints, it is important to capture local connectivity of the state lattice, within a limited radius from the origin.

We use this intuition to define a finite subset of the state lattice that only includes paths from the origin out to some radius. This *control set* is desired to be a minimal set of primitive paths that, when concatenated, can re-generate any other path in the lattice. We seek to make this control set minimal because it is used as a set of alternatives, *outdegree*, at every node during search, and it is important to minimize it for efficiency.

Thus, the control set can be viewed as a time-limited reachability graph. The state lattice is used implicitly to define such a set. An important part of this paper is the discussion on how to construct the set of controls such that it is small in size (to minimize the branching factor of graph search), yet preserves the connectivity of the state lattice.

2 Prior Work

The utility of the lattice is hinged on the assumption that it is possible to determine a feasible path between any two configurations in a C-space without obstacles. While this is itself a very difficult problem, it has been the objective of much research in the past century. Frazzoli et al. in [3] suggest that there are many cases where efficient, obstacle-free paths may be computed analytically, e.g. the systems with linear dynamics and a quadratic cost (double or triple integrators with control amplitude and rate limits). The cases that do not admit closed-form solutions can be approached numerically by solving an appropriate optimal control problem.

A fast nonholonomic trajectory generator was described in [6]. It generates polynomial spiral trajectories, such that a

path is specified by a continuous control function: curvature as a function of path length. Another method for generating continuous curvature paths is presented in [13].

In the case of systems that can be expressed as $\ddot{q} = g(q, u)$ and are not underactuated, it was shown that through careful discretization in control space it is possible to force the resulting reachability graph of system to be a lattice [10]. It was also shown in [12] that this technique can be applied to a large class of nonholonomic systems. That approach presents a way to generate a path given its terminal points and shows how under suitable conditions a regular lattice of reachable points can be achieved. However, this is usually difficult to achieve, and under most quantizations the vertices of the reachability graph are unfortunately dense in the reachable set [10]. By using an inverse path generator, we can choose a convenient discretization in control and state space, one that makes the search more efficient. This also allows us to use continuous control functions that are natural for real systems.

The importance and difficulty of enforcing differential constraints also has a long history [8]. Barraquand and Latombe encoded nonholonomic constraints in the discretized configuration space and incrementally explored that space by discretizing control space, choosing a control and integrating the system kinematics [1]. Similar ideas were utilized in Probabilistic Roadmap (PRM) methods that were very influential in motion planning [4]. However, a recent trend appears to favor more deterministic variants of the PRM [11]. In [2], Quasi-PRM and Lattice Roadmap (LRM) are introduced by using low-discrepancy Halton/ Hammersley sequences and a regular lattice, respectively, for sampling. These approaches were shown to make planning more efficient by more uniform sampling of the state space and achieved more efficient exploration. Both methods were shown to be resolution-complete; the LRM appeared especially attractive due to its properties of optimal dispersion and near-optimal discrepancy.

Also, a "Lazy" variant of these methods was discussed that avoided collision checking during the roadmap construction phase. In this manner the same roadmap could be used in a variety of settings, at the cost of performing collision checking during the search. An even "lazier" version is suggested, in which "the initial graph is not even explicitly represented" [2]. In this regard, our approach of using an implicit lattice and searching it by means of a pre-computed control set that only captures local connectivity is very similar to the Lazy LRM. Our contribution is in exploring the conjecture made in that work and successfully applying it to nonholonomic motion planning.

In the development of Rapidly-exploring Dense Trees (RDT) for motion planning with differential constraints, the importance of designing off-line a family of motion primitives that captures the specifics of the system under consideration is noted [10]. In this light, our proposed control set is precisely that set of primitives that reflects symmetries of wheeled vehicles and encodes nonholonomic constraints. Thus, this work is aligned with the latest in developments of RDT and new deterministic sampling methods developed for them, while borrowing the efficiencies of "Lazy" exploration of state space.

Initial concepts of this work were explored in a successful field implementation of a nonholonomic motion planner built using the state lattice of limited size represented explicitly [5].

3 State Lattice

In order to proceed with the construction of the control set, it is helpful to explore the properties of the search space that it is designed to represent, i.e. the implicit state lattice.

3.1 Control Model For a Car-Like Robot

We consider a car-like robot as a 4D system from a kinematic perspective and as a 3D system from a geometric perspective. We assume (x, y) to denote the reference point of the vehicle, θ its orientation, and ζ the angle of steering wheels. The configuration space is $C = \tilde{R}^2 \times (S^1)^2$. The two controls of the car are its velocity v and the time derivative ω of the steering angle. A configuration $q = (x, y, \theta, \zeta)$ is considered *admissible* if $|\zeta| \leq \zeta_{max}$, i.e. the steering angle is constrained by some mechanical bounds [7]. Thus, a car-like vehicle obeys the following control model.

$$\begin{bmatrix} x \\ y \\ \theta \\ \varphi \end{bmatrix} = \begin{bmatrix} \cos\varphi\cos\theta \\ \cos\varphi\sin\theta \\ \sin\varphi \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega \qquad (1)$$

This system was proven in [9] to be small-time controllable at any point. Thus, starting from any configuration, the set of configurations reachable by moving with bounded velocity in finite time always contains a neighborhood of the initial configuration. This result enables us to consider some finite neighborhood of the state lattice, a control set, to be a good representative of all motions that are reachable in finite time.

An important class of wheeled robots, such as differential drive, obey a more general system definition. By setting $\tilde{v} = v\cos\zeta$ and $\omega = v\sin\zeta$ we obtain the following 3D control system:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \tilde{v} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tilde{\omega} \qquad (2)$$

where $\tilde{v}$ and $\tilde{\omega}$ can be considered as translational and angular velocity, respectively [7]. By constraining these two variables, this system formulation can be utilized for car-like robots, so henceforth we use it in our discussion.

For path planning purposes it is convenient to relate the two velocities by

$$\kappa = \frac{\tilde{\omega}}{\tilde{v}} \qquad (3)$$

where κ is curvature of the path. Curvature is defined for all motions of non-zero translational velocity, and thus is a convenient representation of paths.

3.2 Inverse Path Generation

Among several approaches discussed in Section 2 that allow finding a sequence of controls from a given initial configuration to a final configuration, we evaluated the one described in [6]. This approach allows fast generation of nonholonomic trajectories. The assumed form of the solution path is a curvature polynomial of arbitrary order. The

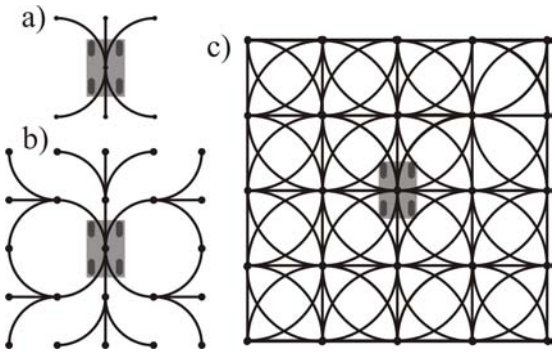

Figure 1: Reachability graph for the Reeds-Shepp Car. Various stages of construction of the reachability graph are shown. A Reeds-Shepp Car of three actions specially discretized is allowed to move anywhere in the workspace.

method was shown to provide good results and in principle allows optimization w.r.t. several criteria. In particular, a convenient feature was that computed paths contain the least curvature variation. The continuous specification of paths was convenient to manipulate and execute in vehicle controllers. The method executes practically in real-time: a query is computed in about 1 millisecond.

3.3 Constructing the State Lattice

To elucidate our idea of the state lattice, we give an example of constructing it for the Reeds-Shepp Car.

We illustrate building the lattice by noting its similarity to a reachability graph of a system. Figure 1 shows a reachability graph for the Reeds-Shepp Car with three actions, suitably discretized for clarity. As the car travels infinitely, the graph is constructed by "compiling" all feasible motions.

The state lattice, however, is constructed by using the inverse path generator to find paths between any node in the grid and the origin (black square in Figure 2). Although for this simple example, the lattice structure can be achieved without using inverse path generation (i.e. by choosing a special discretization of state and control space [10]), this is much more difficult to do in the general case of wheeled robots. Since the choice of the origin in constructing the lattice was arbitrary (because the same set of motions is available from any node) we can copy the entire set of feasible paths to any node in the lattice (we will formally define this in Section 3.5).

In the limit, as both the graph in Figure 1 is expanded by the Reeds-Shepp car traveling infinitely long, and the lattice in the Figure 2 is built by including the feasible paths from the origin to any node (and copying the resulting set to all nodes), both the graph and the lattice will be identical by virtue of representing all feasible paths (Figures 1c and 2d). Without loss of generality, we henceforth consider the state lattice to be a valid representation of the system's reachability graph.

Next we look more closely at two important properties of the lattice and its utility for formulating nonholonomic path planning as graph search.

3.4 Discretization

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.

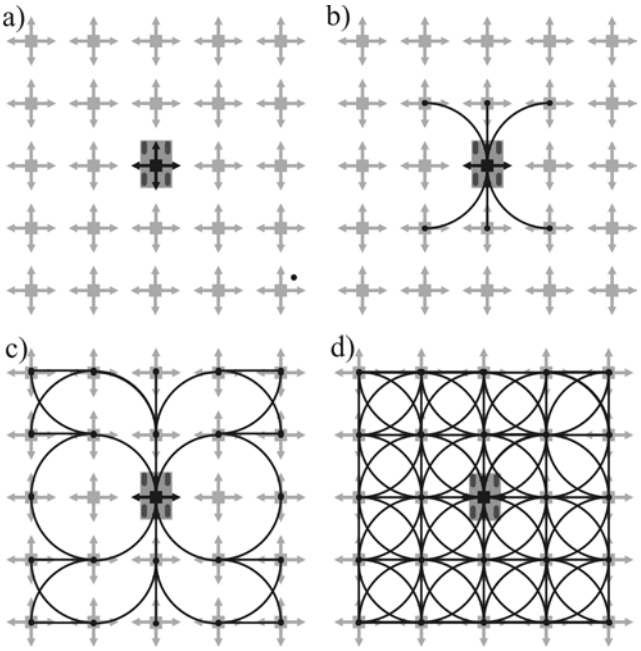

Figure 2: Constructing the lattice for the Reeds-Shepp Car. In a) we define a discretization in C-space (an (x, y) grid is chosen here, arrows indicate allowed headings), an origin is chosen; b) for 8 neighbor nodes around the origin, feasible paths are found; c) same query is extended outward to 24 neighors, only a few direct paths shown; d) complete lattice.

The states are the nodes in the lattice and the motions that connect the states are the edges.

While the state vector can certainly have arbitrary dimension, for this paper we have implemented the state lattice in 4 dimensions. Each node of the lattice therefore represents a 4-dimensional *posture* that includes 2D position, heading and curvature. This representation allows the lattice to be general enough to satisfy typical nonholonomic and holonomic constraints for automobiles.

We assume that position is discretized into rectangular grid of separation Δ_l. Any useful heading discretization is admissible, and in Section 5.1 we introduce a heading discretization scheme that leads to maximizing straight line motion plans. For now, let us assume uniform division of the interval $[0, 2\pi]$. The same holds for discretizing curvature in the interval $[-\kappa_{max}, \kappa_{max}]$, where

$$\kappa_{max} = \frac{1}{R_{min}} \qquad (4)$$

is maximum curvature, a reciprocal of R_{min}, minimum turning radius, expressed in the same units of length as Δ_l. We assume that this value is defined for a planning problem.

3.5 Regularity

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. A discretization exhibiting some degree of regularity leads to a set of motion options which is similarly regular.

For the balance of the paper, we adopt the assumption that the state space discretization is regular in at least the trans-

lational coordinates (x, y). Specifically, if the path between two postures:

$$[x_1, y_1, \theta_1, \kappa_1] \rightarrow [x_2, y_2, \theta_2, \kappa_2]$$

is feasible, then so is the path

$$[x_1 + n\Delta_l, y_1 + n\Delta_l, \theta_1, \kappa_1] \rightarrow [x_2 + n\Delta_l, y_2 + n\Delta_l, \theta_2, \kappa_2]$$

for any value of the integer n. While the starting and ending states for two such motions are distinct, the motion itself (perhaps encoded as a steering function) is not. Indeed, we can now consider only the set of motions emanating from the origin. Only that canonical set needs to be represented explicitly. The entire lattice can be obtained by repeating this representation at any other node (Fig. 2b).

3.6 Equivalence of Paths in the Lattice

State space discretizations associate small regions of space surrounding the center of a cell with the cell itself. If space is to be discretized, it is consistent to ask whether paths through space should be similarly discretized.

We will consider two paths (with identical endpoints) which are sufficiently far apart to be distinct. If the paths are "sufficiently" close together, we can consider them to be equivalent. We define a path p_2 to be equivalent to p_1 if the path p_2 is contained in the interior of a certain region Q around p_1, defined as a set of configurations within a certain distance of p_1:

$$\forall q \in p_1, \forall q' \in C, Q = \langle q' | \rho(q', q) < \delta_e \rangle \quad (5)$$

given some metric ρ and constant δ_e.

All paths that satisfy this criterion are considered to belong to the same equivalence class (Figure 3).

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 exploiting this error, motion planning can be made more efficient, as presented below.

3.7 Lattice as a Search Space

The state lattice presented thus far possesses the properties necessary to express nonholonomic motion planning as graph search.

A cost-map can be overlaid on the lattice to represent obstacles or other criteria with respect to which we want to find an optimum obstacle-free motion plan (energy, slope hazard, etc.). We also assume a path sampling procedure that returns the cost of traversal of a path given the cost of

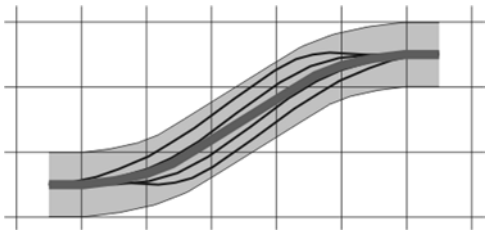

Figure 3: Path Equivalence. A variety of paths between two configurations (thin black lines) that are contained in the boundary (blue lines) are considered to be equivalent and represented by a canonical path (red line).

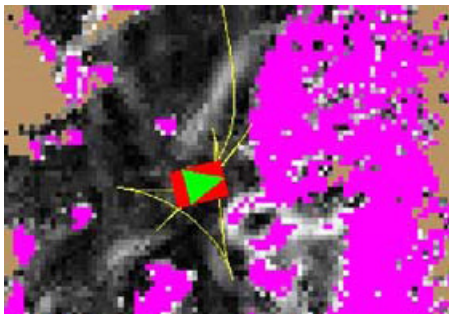

Figure 4: A non-holonomic planner based on the state lattice. A 5-point turn maneuver is generated by the state lattice based planner to avoid a natural cul-de-sac.

cells spanned by the path. By virtue of containing all feasible motions, the lattice is a cyclic graph. Any standard systematic heuristic graph search algorithm can be applied.

An implementation of a nonholonomic motion planner based on the state lattice has been successfully constructed for an off-road robotics application [5]. This application of generating complicated start-stop motions in dense tree mazes motivated the original investigation into a regular state lattice encoding constraints directly.

In that work, a further simplification was made in considering only a unique path between two lattice nodes. Also, only a finite subset of the lattice was considered. Despite the latest computing hardware on-board the robot, the subset of the lattice that could be represented and processed (to produce plans in 20 seconds) allowed motion plans within a rather limited area around the robot.

Nevertheless, the real-time implementation of the planner was shown to work very well for navigating a mobile robot in natural cluttered environments. Figure 4 depicts an example motion plan that allowed the vehicle to avoid a natural cul-de-sac. The greyscale portion in the figure represents the cost-map, pink indicates obstacles, and orange is the area of unknown cost. Yellow line represents the generated plan.

4 Control Set

Whereas the state lattice was used as a search space for a motion planner, there are considerable limitations to this approach in the case of planning over a large area in a workspace. Memory requirements for storing the entire lattice and the number of alternatives a search algorithm must consider (outdegree) impose a significant limit to applicability of this method.

Similar issues arise in applications of Lazy LRM to nonholonomic motion planning. In many off-road scenarios it is desired to consider a large area of the workspace during planning, which results in a prohibitively large explicit representation of the roadmap. Hence there is special value in considering even a "lazier" version that considers the underlying lattice implicitly, which is what we advocate here.

Also, the majority of the literature on PRM and its variants considers a rather arbitrary choice of the neighborhood in order to pick to expand a node. Often a constant radius around a node that is to be expanded is used. Here we present a principled method of choosing the neighborhood in question that both offers practical guarantees of best

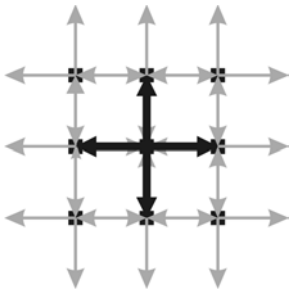

Figure 6: Isolating a minimal set of motions in a 4-connected grid. The minimal set of motions, a control set, can be used to re-generate the entire grid.

exploration of the lattice and keeps the neighborhood size small to preserve search efficiency.

4.1 Path Decomposition

With the insights obtained in Section 3.6, we again look at the lattice as a concept derived from a grid. The regularity property of the grid implies that it is possible to isolate a certain representative set of connections which is repeated everywhere in the grid. As is illustrated in Fig. 6, for the case of a 4-connected rectangular grid, it is easy to identify a minimal set of connections. If we cast the grid in the context of motion planning, we understand connections as elementary motions between two nodes. This minimal set of connections is the (finite) set of controls that is identical for every state and that allows us to generate arbitrarily long motion plans in the infinite grid. This concept has been used in motion planning for some time [8].

In a similar fashion, if we could identify such a control set for a lattice, we could use it to address the computational issues mentioned above and essentially create a finite representation of the lattice.

By invoking the notion of path equivalence class and some non-zero δ_e, we can substitute a path with two other paths such that their concatenation generates a motion that belongs to the same equivalence class as the original path. We define *path decomposition* as the problem of finding two such constituents of a path (Figure 7).

By definition of path decomposition, the two constituent paths must meet at a lattice node. Intuitively, the longer a path is, the more lattice nodes it comes "close" to, hence the easier it is to find a decomposition because there are more "opportunities" to do so. Hence, it may be possible to decompose all motions in the lattice and create a finite control set, as was done for a grid. However, whereas in case of the grid, the nodes are equidistant and hence component paths have constant length, no such assumptions can be made regarding paths in the lattice. Due to this generality, it is difficult to create a rigorous proof that the entire infinity of motions, of infinite length, of the lattice can be decomposed in this fashion.

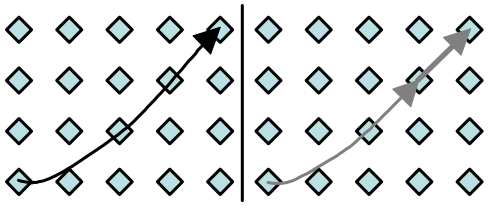

Figure 7: An illustration of path decomposition.

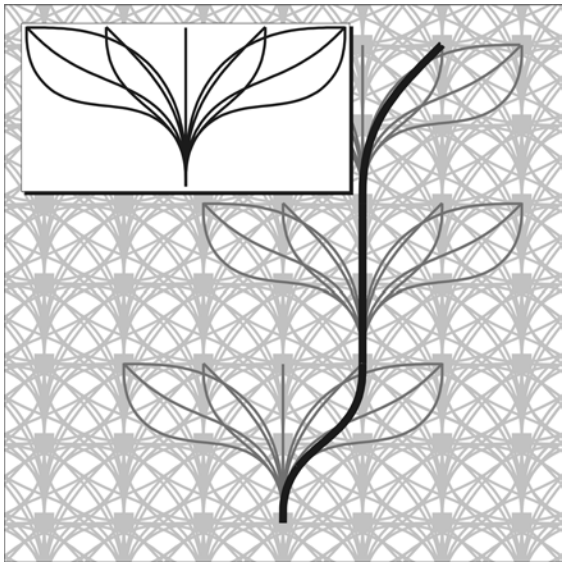

Figure 8: An illustration of identifying a control set in the lattice. The same set of motion options (top left corner) is centered at every node. This set is repeated at the nodes in the lattice in order to generate the path (thick black curve).

However, through a simulation study we concluded that this is possible for realistic vehicle parameters (κ_{max} and path following error δ_e). We considered over 2000 different (relatively long) paths in the lattice and showed that all of them could be decomposed into at least two (usually more) smaller paths.

Thus, the control set allows us to eliminate redundancies of the lattice both in terms of the variety of paths between nodes (through the notion of path equivalence), and in terms of generally unlimited path length (path decomposition).

4.2 Properties of the Control Set

As a representation of the state lattice as a search space, the control set contains motion alternatives that a search algorithm has to consider at each node. There are several important properties of the control set that make it attractive for nonholonomic motion planning.

4.2.1 Minimal Set of Feasible Motions

The process of path decomposition can be implemented in a variety of ways. Our formulation of decomposition admits constructing a decomposition algorithm that generates the smallest possible set of motion alternatives. That is, given some control set, if there exists another set that has fewer motion alternatives, then the algorithm chooses it. Inductively, the algorithm will arrive at the smallest representation of motions of the lattice. This result implies that the resulting control set is the most efficient search space satisfying the original constraints applied to its construction. However, given that we make no assumptions about the nature of paths, verifying that a control set is indeed minimal requires comparing it to all possible templates, which is intractable.

Nevertheless, in Section 5 we discuss control set generation algorithms that operate on a finite, but large, subset of the lattice, use realistic vehicle parameters, and generate near-minimal motion templates. Such templates were shown to

preserve the valuable quality of being able to re-generate, through concatenation, any feasible path in the lattice.

4.2.2 Control Set Radius

We define control set radius, denoted as R_{cs}, as a measure of the size of the control set. If we visualize a smallest square outline (bounding box) that encloses 2D workspace projections of all paths in the control set, then R_{cs} denotes half of the length of the squares sides.

4.2.3 Relative Minimum Turning Radius

An interesting property of the control set is that it couples the vehicle-dependent parameter, κ_{max}, and resolution of the lattice, Δ_l. Thereby, the size of the control set is related to maneuverability of the vehicle in terms of Δ_l. To capture this relationship, we define the notion of relative minimum turning radius, R':

$$R' = \frac{R_{min}}{\Delta_l} = \frac{1}{\kappa_{max}\Delta_l} \qquad (6)$$

We use this notion later as a scale-invariant characteristic of a control set (i.e. the measure of "curviness" of its paths).

5 Generation of the Control Set

Here we provide an example of generating and using control sets in practical scenarios.

5.1 Inverse Generation

Given a method to generate the set of distinct feasible paths to a single state, the control set can be generated by a process of structured elimination. First, paths to all states one unit from the origin are generated, then, paths to all states two units from the origin, etc. When a path is considered, it is tested for passing sufficiently close to an intermediate state, and if so, it is removed from the control set because it can be decomposed into the path to this state from the origin and the path from this state to the end-point. Since we are moving radially outward, any path that can be decomposed may be removed from the control set because its "ingredients" have already been considered. Each of them is either in the control set already or does not need to be because it itself is decomposable.

This process terminates at the certain radial distance from the origin when all paths at that distance can be decomposed. Through simulation studies similar to the one mentioned in Section 4.1 we verified that this termination condition is a good heuristic for obtaining a control set that spans the entire state lattice.

This inverse generation strategy was implemented for an Ackerman steer vehicle. We discuss several important

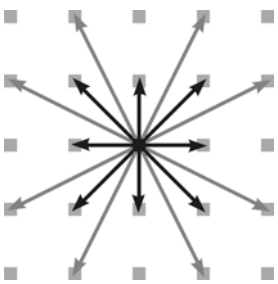

Figure 9: Heading discretization. A discretization of heading is chosen such that we implicitly represent straight paths whenever it is possible.

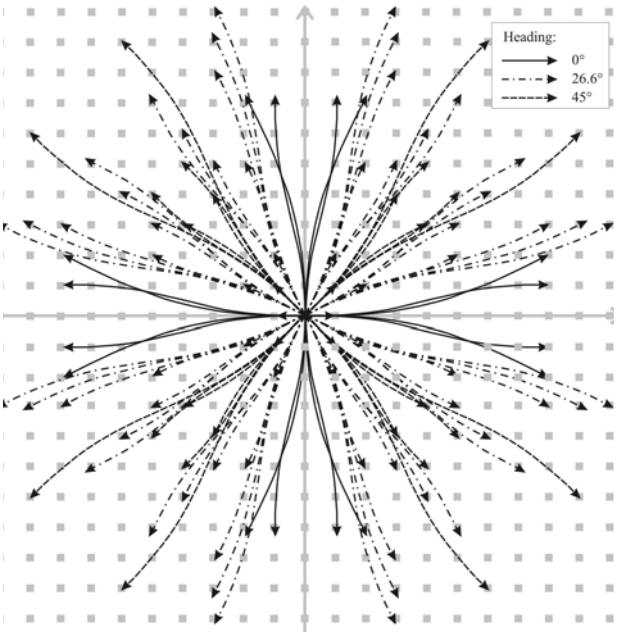

Figure 10: An Example Control Set. This set of paths was generated at $R' = 20$. Paths in this set can be combined to generate any other spatially distinct feasible motion in the state lattice emanating from the origin at the initial heading and curvature. Only three sets of paths with initial heading of 0, 26.6 and 45 deg. are specified; all others are obtained by reflection around x- and y-axes, and the two diagonals.

design decisions and the results of this control set generation.

Although a uniform discretization of heading is an option, we found it useful to discretize heading differently. Discrete headings that point directly to another state encode straight lines in many more directions than equally spaced headings (Fig. 9).

Motion alternatives were expressed as polynomial steering (curvature) functions of cubic order:

$$\kappa(s) = a + bs + cs^2 + ds^3 \qquad (7)$$

Such functions possess exactly the 5 degrees of freedom required to join any two poses with arbitrary initial and terminal curvatures. An algorithm to generate the steering function between two arbitrary states is described in [6]. This choice also implies that the computed path between any two given states with specified curvatures is unique.

It was confirmed that, given choices of R' (similar to values used in a previous mobile robot project [5]), there exists a control set radius beyond which all trajectories pass sufficiently close to some intermediate state and are decomposed. Table 1 presents some experimental values of the generated control sets. Also, we include the time required to generate each set, which appears to be quite reasonable for an off-line process that can be utilized in a variety of motion planning scenarios.

5.2 Experimental Results

We used the control set generated in the previous section to solve an instrument placement problem for a planetary rover (Figure 11). Here the environment is very cluttered as it includes a variety of difficult terrain on the slope of the crater; the background is a pseudocolor of ground height,

such that the slope is downward from right to left. The rover must approach each target at a specified heading. Two paths are shown for each of the five targets: actual paths (solid lines) and one of other consider paths (dotted lines). This illustrates how the planner takes into account energy considerations. Selecting lower curvature path is important, but going over hills (e.g. in the middle of figure) has higher cost.

The planner was able to handle this problem quite easily. The experiment was performed 100 times, and the average run-time was 3.5 seconds for each path segment.

6 Applications

A fast and efficient implementation of heuristic search can be built using the control set to search the space of feasible motions (the state lattice) in a systematic fashion.

In the discussion we focused on applying this technique to wheeled mobile robots. However, the concept of the control set is portable in principle to any motion planning problem.

The presented method can be viewed as a steering planning component that can be combined with a velocity planning component to create a kinodynamic motion planner, similar to [14]. If the capability of the vehicle to follow a path at high speed can be roughly related to a maximum angular velocity the vehicle can experience, then the maximum velocity of traversal at any point along the path can be directly obtained from path specification as $\kappa(s)$ using (3). This is in accord with the intuition that the curvier the path, the slower a vehicle must traverse it.

Table 1Experimental Results of Generating Control Set

R'	outdegree	R_t	time
20	12	15	20 min.
10	12	13	11 min.
5	9	10	4 min.

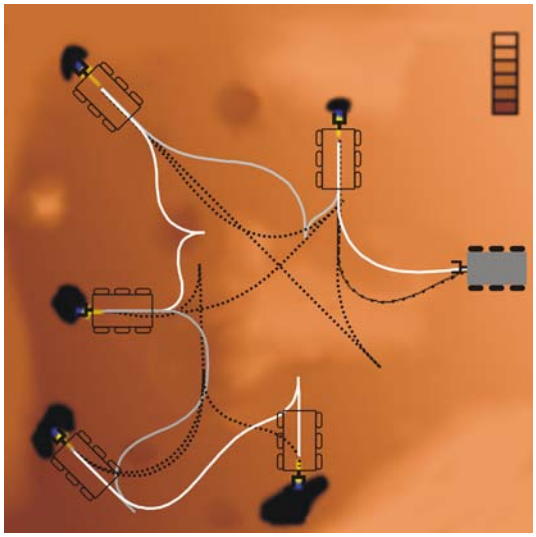

Figure 11: Example application. Here a planetary rover instrument placement problem is solved. The rover must approach five science objects at specified heading in cluttered environment on the slope of a crater.

7 Conclusions and Future Work

This work has proposed a generative formalism for the construction of motion templates for constrained motion planning. The inherent encoding of constraints in the resulting representation re-renders the problem of motion planning in terms of unconstrained heuristic search. As the outdegree of nodes becomes comparable to that of a grid, similar efficiencies can be expected in the generation of plans of equal numbers of states. Also, the encoding of constraints is an offline process that does not affect the efficiency of on-line motion planning.

Ongoing work includes designing a motion planner based on dynamic heuristic search which would allow it to consider arbitrary moving obstacles, the extension of trajectory generation to rough terrain, and hierarchical approaches which scale the results to be applicable to kilometers of traverse.

8 References

[1] J. Barraquand and J.-C. Latombe, “On nonholonomic mobile robots and optimal maneuvering,” in *Proc. of the IEEE Int. Symp. on Intelligent Control*, 1989.

[2] M.S. Branicky, S. LaValle, S. Olson and L. Yang, “Quasi-randomized path planning,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, May 2001.

[3] E. Frazzoli, M.A. Dahleh, and E. Feron, “Real-time motion planning for agile autonomous vehicles,” in *Proceedings of the American Control Conference*, Arlington, VA, June 2001.

[4] D. Hsu, R. Kindel, J.-C. Latombe and S. Rock, “Randomized kinodynamic motion planning with moving obstacles,” *Int. J. of Robotics Research*, vol. 21(3), pp. 233-255, March 2002.

[5] Kelly, A., et al., “Toward Reliable Off-Road Autonomous Vehicle Operating in Challenging Environments”, International Symposium on Experimental Robotics, June, 2004, Singapore.

[6] Kelly, A., Nagy, B. "Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control". *International Journal of Robotics Research*, 22, (7-8), 2003.

[7] F. Lamiraux and J.-P. Laumond, “Smooth motion planning for car-like vehicles,” *IEEE Transactions on Robotics and Automation*, vol. 17(4), pp. 498-502, August 2001.

[8] Latombe, J.-C. *Robot Motion Planning*. Kluwer Academic Press, Boston, 1991.

[9] J.-P. Laumond, S. Sekhavat and F. Lamiraux, “Guidelines in nonholonomic motion planning,” in *Robot Motion Planning and Control*, J.-P. Laumond, Ed., New-York: Springer-Verlag, 1998.

[10] S. LaValle, *Planning Algorithms*, online: http://msl.cs.uiuc.edu/planning .

[11] S. LaValle, M. Branicky, and S. Lindemann, “On the relationship between classical grid search and probabilistic roadmaps,” *Int. J. of Robotics Research*, vol. 23(7-8), pp. 673-692.

[12] Pancanti, S., et al. “Motion Planning through Symbols and Lattices”. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, 2004.

[13] Scheuer, A., Laugier, Ch. “Planning Sub-Optimal and Continuous-Curvature Paths for Car-Like Robots”. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 25-31, October 1998.

[14] Wang, D., Feng, Q., “Trajectory Planning for a Four-Wheel-Steering Vehicle”. *In Proc of the IEEE Int. Conf. on Robotics and Automation*, May 2001.