Improving the Performance of Sampling-Based Planners by Using a Symmetry-Exploiting Gap Reduction Algorithm

Peng Cheng Emilio Frazzoli Steven M. LaValle

University of Illinois
Urbana, IL 61801 USA
{pcheng1, frazzoli, lavalle}@uiuc.edu

***Abstract*— Although sampling-based planning algorithms have been extensively used to approximately solve motion planning problems with differential constraints, gaps usually appear in their solution trajectories due to various factors. Higher precision may be requested, but as we show in this paper, this dramatically increases the computational cost. In practice, this could mean that a solution will not be found in a reasonable amount of time. In this paper, we substantially improve the performance of an RRT-based algorithm by planning low precision solutions, and then refining their quality by employing a recent gap reduction technique that exploits group symmetries of the system to avoid costly numerical integrations. This technique also allows PRMs to be extended to problems with differential constraints, even when no high-quality steering method exists.**

I. Introduction

Motion planning problems with differential constraints include both kinodynamic motion planning and nonholonomic motion planning problems, in which constraints exist for both the configuration and its time derivatives. The final objective is to design an open-loop piecewise continuous control history which will safely drive the robot system from an initial state to a goal state. A challenging example is shown in Fig. 1, in which a control history is designed to maneuver a spacecraft close to a large space structure, after the failure of some of the on-board thrusters. Because of pointing constraints on the communication antenna, on-board telescope and star tracker, the spacecraft is under restrictive constraints both on its position and its attitude. There exists strong evidence [6], [7], [8] that motion planning with differential constraints is intractable. Therefore, many sampling-based methods have been designed. This includes incremental search algorithms (ISAs) (e.g., RRTs [27]) and roadmap methods (e.g., PRMs [38]).

One approach is to first find a path using a holonomic planner, and then time-parameterize the path and design the control [4], [17], [28], [39]. Because the dynamics are ignored in the first step, the returned path might not be executable or efficient. Another approach is an incremental search algorithm (ISA). Since dynamics constraints are considered in the planning process, the returned solutions are guaranteed to be executable [1], [3], [5], [12], [13], [14], [15], [20], [26], [29], [36], [30], [34]. The majority of robot systems involve continuous time and inputs; therefore, most ISAs first discretize the space of piecewise continuous control histories into a finite subset, which decreases the problem complexity. A directed search graph is incrementally built, which corresponds to family of partial solution trajectories. Initially, there are a small number of nodes. In each iteration, a node in the graph is chosen and extended to generate new nodes and edges. If a path in the graph that connects the initial state and a goal state exists, a solution is returned. PRMs have also been used for systems with differential constraints [38], [40] when an analytical steering method is available.

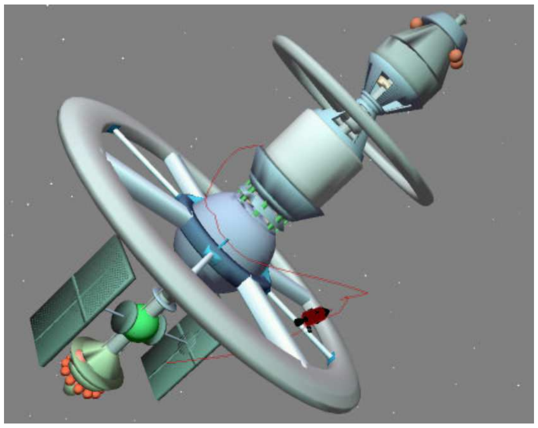
Fig. 1. This extremely challenging problem requires trajectory design for a crippled spacecraft that has complicated orientation constraints (details are in Section VI) and must drift among obstacles.

One common problem with ISAs and other sampling-based planning algorithms is that there exist gaps in their solutions since the solution control history might not exactly drive the system between two given states. One main reason is the input space discretization. Under the discretization, some systems are not small-time locally controllable (STLC): the sets of reachable states have the structure of a lattice, which prevents the exact matching of the trajectory endpoint with the goal state in a finite number of steps. In cases in which the set of reachable states is everywhere dense [31], or even continuous [14], it is possible in principle to add a sequence of controls to move the final point arbitrarily close to the end goal, but this is done at the expense of the efficiency of the trajectory.

Gaps greatly degrade the quality of the solutions, especially when a gap appears far from the end because a small offset early in the trajectory might greatly change the final state after integration. Many ISAs can quickly return low precision solutions with big gaps. However, finding high precision solutions with small gaps usually increases the running time dramatically. Sometimes, if the set of reachable states has a lattice structure under a control space discretization, a solution will not be returned by an ISA if the distance between nearest reachable states to any goal state is larger than the given tolerance. In this case, a higher resolution discretization of the control space might be necessary, which also causes longer running time.

If a separate algorithm could efficiently reduce gaps, then ISAs can find high precision solutions faster by first finding low precision solutions and converting them into high precision solutions. One way to reduce gaps is to use steering algorithms [5], [33], [23], [25], [37]. Steering methods are only available for few classes of systems, such as kinematically controllable systems [5] and differentially flat systems [37]. Furthermore, obstacle constraints are difficult to incorporate explicitly in steering methods. Another problem in our context is that the cost of the resulting trajectories might be dramatically increased at the gaps. Instead of steering to eliminate gaps in a control history, we have developed a method [9] that exploits geometric properties of dynamical systems to reduce gaps by efficiently perturbing the control history. Besides our work, avoiding obstacles by optimizing computed paths for nonholonomic systems is presented in [24]. In this paper, we design new planners by combining an RC-RRT [10] with the gap reduction method, and use the new planner to solve several challenging motion planning problems with differential constraints. Also, we show how the gap reduction idea can be used to construct a PRM [19] when a steering method is not available.

II. The Discretized Motion Planning Problem

In this section, we first define the original motion planning problem we wish to solve, and then define a discretized version, used for sampling based planners.

Motion planning with differential constraints: A motion planning problem with differential constraints, denoted as $\mathcal{P}$, is a tuple, $(\mathbf{X}, \rho, \mathcal{D}, x_{\text{init}}, X_{\text{goal}}, \mathcal{U}, f)$. The state space, $\mathbf{X}$, is a compact differentiable manifold of dimension n with a metric function ρ. The constraint function $\mathcal{D}$ is a map from $\mathbf{X}$ to $\mathbb{R}$ and $\mathcal{D}(x) \leq 0$ means that the state $x \in \mathbf{X}$ is violation free. The initial state is x_{init}, and the set of goal states is X_{goal}. The input space $\mathcal{U} \subset \mathbb{R}^m$ is compact, and $m \leq n$. The dynamics of the system, assumed to be time-invariant, are described by a set of Ordinary Differential Equations (ODE) of the form

$$\dot{x}(t) = f(x(t), u(t)), \tag{1}$$

in which $x(t) \in \mathbf{X}$, $u(t) \in \mathcal{U}$, and $\dot{x}(t)$ denotes the time derivatives of the state.

Consider a piecewise continuous open-loop control history $\pi : [0, t_f) \to \mathcal{U}$. Let $\Phi_\pi : \mathbf{X} \times [0, t_f) \to \mathbf{X}$ denote the state transition map for (1) under the action of π, i.e., $\Phi_\pi(x_0, t)$, with $t \in [0, t_f)$, is the solution of the initial condition problem $\dot{x}(t) = f(x(t), \pi(t))$, with $x(0) = x_0$. Clearly, $\Phi_\pi(x_0, 0) = x_0$.

Discretized motion planning problems: Most sampling based planners restrict the search for a solution to the motion planning problem by discretizing time and the control inputs. In other words, given a sampling interval Δt, and a finite set $\tilde{\mathcal{U}} \subset \mathcal{U}$, solutions are parameterized by a finite array of control values $\tilde{\pi} = (\tilde{\pi}_0, \ldots, \tilde{\pi}_{k-1}) \in \tilde{\mathcal{U}}^k$, for some $k \in \mathbb{N}$, in the sense that $\pi(t) = \tilde{\pi}_{\lfloor t/\Delta t \rfloor}$. This transforms $\mathcal{P}$ into a discretized motion planning problem $\tilde{\mathcal{P}} = (\mathbf{X}, \rho, \mathcal{D}, x_{\text{init}}, X_{\text{goal}}, \tilde{\mathcal{U}}, \Delta t, f)$.

We say that the policy π is an exact solution to $\mathcal{P}$ if the following conditions are satisfied: 1) $\mathcal{D}(\Phi_\pi(x_{\text{init}}, t)) \leq 0, \forall t \in [0, t_f)$, and 2) $\exists t_{\text{goal}} \in [0, t_f) : \Phi_\pi(x_{\text{init}}, t_{\text{goal}}) \in X_{\text{goal}}$. Since (finite-length) exact solutions do not exist in general for the discretized problem, approximate solutions must be considered in that case. We say that the control policy π—or its discrete parameterization $\tilde{\pi}$—is a solution with tolerance ϵ_π if: 1) $\mathcal{D}(\Phi_\pi(x_{\text{init}}, t)) \leq 0, \forall t \in [0, t_f)$, and 2) $\exists t_{\text{goal}} \in [0, t_f), x_{\text{goal}} \in X_{\text{goal}} : \rho(\Phi_\pi(x_{\text{init}}, t_{\text{goal}}), x_{\text{goal}}) < \epsilon_\pi$. In the remainder of the paper, we will often use the shorthand notation $\Phi_\pi^t(x_0) := \Phi_\pi(x_0, t)$.

III. Sampling-Based Planners and Solution Gaps

We will first analyze the ISAs to see how gaps are generated. Then gaps in roadmap based planners are discussed.

Incremental search algorithms: The general form of ISAs is shown in Fig. 2. An ISA will take $\tilde{\mathcal{P}}$ and ϵ_s as parameters and iteratively build a search graph for the solution, in which ϵ_s is the gap tolerance. To obtain a solution with tolerance ϵ_π, ϵ_s is normally much smaller than ϵ_π (A quantitative relationship between ϵ_s and ϵ_π is conservatively estimated in [10]). The search graph is a directed graph $G\langle N, E\rangle$, in which each node n in N is associated with a state $x(n) \in X$, and each edge $e \in E$ is associated with an input $u(e) \in \tilde{\mathcal{U}}$.

In Line 1, the search graph G is initialized with any starting states. In a single-directional search method, G initially contains a single node, n_{init}, associated to x_{init}. For a bidirectional search method, G also contains n_{goal}, associated to x_{goal}. Similarly, more initial nodes could also be added. Line 3 selects a node, $n_{\text{cur}} \in N$, and determines the control input $U_{n_{\text{cur}}} \in \tilde{\mathcal{U}}$ for extension. Line 4 selects a state, x_{int} that may serve as an intermediate goal. A local planner chooses from $U_{n_{\text{cur}}}$ an input, u_{new}, which drives the system from $x(n_{\text{cur}})$ toward x_{int} over a time interval Δt, generating the new state x_{new}. Line 6 determines whether the new trajectory portion satisfies the constraints (this could alternatively be accomplished

```
INCREMENTAL_SEARCH($\tilde{\mathcal{P}}, \epsilon_s$)
1   $E \leftarrow \emptyset$; $N \leftarrow$ some starting states;
2   repeat
3       $n_{\rm cur}, U_{n_{\rm cur}} \leftarrow SelectCurrentNode(G)$;
4       $x_{\rm int} \leftarrow SelectIntermediateGoalState(G)$;
5       $u_{\rm new}, x_{\rm new} \leftarrow LocalPlan(x(n_{\rm cur}), x_{\rm int}, U_{n_{\rm cur}}, \Delta t)$;
6       if $u_{\rm new}$ leads to a violation-free trajectory then
7           $G.InsertNode(x_{\rm new})$;
8           $G.InsertDirectedEdge(n_{\rm cur}, n_{\rm new}, u_{\rm new})$;
9           $SolutionCheck(n_{\rm new}, \tilde{\mathcal{P}}, G, \epsilon_s)$
10  until TerminationCondition(G)
11  return Failure;
```

Fig. 2. A general template for incremental search algorithms.

in the local planner). If so, then a new node with state $x_{\rm new}$ and a new edge with input $u_{\rm new}$ are added to G. In Line 9, whether a solution exists will be checked against $x_{\rm new}$. If $\rho(x(n_{\rm new}), x(n)) < \epsilon_s$ for some $n \in N$, then $n_{\rm new}$ is identified with n in G. Furthermore, if there is a directed path in G passing $n_{\rm new}$ from $n_{\rm init}$ to $n_{\rm goal}$, the control sequence in the path will be returned as a solution. Otherwise, the process is iterated, until a solution is found or a termination condition is met, at which point the ISA has failed to find a solution. This might occur, for example, after a specified number of iterations.

Gaps in the solutions: For an edge $e \in E$ connecting n_1 and n_2 in N, we say that there exists a gap in e if $\Phi^{\Delta t}_{u(e)}(x(n_1)) \neq x(n_2)$, in which $u(e)$ is the control associated with the edge. When a solution of $\tilde{\mathcal{P}}$ contains a policy which is associated with an edge with a gap, we say that a solution has a gap. Gaps could be induced by many reasons, such as state space discretization, control space discretization, local planners, and numerical integrations. If the state space is discretized into small voxels [2], all states in one voxel will be represented by only one state such that gaps are induced between these states. Gaps also happen when an edge is associated with a control which is designed by a non-exact local planner and does not exactly connect two states. Similarly, numerical integration causes gaps since its results are only an approximation of the actual value. To be concise, only gaps induced by the gap tolerance at Line 9 are considered in the paper. However, methods for this type of gaps could easily be extended to other types of gaps.

ISAs normally are single- or bi-directional tree based search. There are no gaps in edges in the trees because one end state of an edge is obtained by integrating the control associated the edge from the other end state. The only gap happens when we check the existence of a solution with ϵ_s. Take a single directional ISA for example, the gap of a solution π only exists between $\Phi^{t_f}_{\pi}$ and some $x_{\rm goal}$ in $X_{\rm goal}$. Bidirectional ISAs will have only one gap in the middle of a solution.

Roadmap based planners: For roadmap based methods, the roadmap is built by using the local planner to connect states associated with milestones. If only approximate local planners are available, the gap tolerance ϵ_s is used to check the connectivity of those states, i.e., if a control $u : [0, t_u) \rightarrow \mathcal{U}$ is designed by an approximate local planner to connect state x_1 and x_2 and $\rho(\Phi^{t_u}_u(x_1), x_2) < \epsilon_s$, x_1 is considered to be connected to x_2 by u. In other words, a gap might be induced in each edge in the roadmap. There could exist more than one gaps in a solution when the solution path passes through several edges in the roadmap. Generally, bidirectional and roadmap based methods are much more efficient than single directional methods since they have less search depth and more chance to detect solutions. However, especially for roadmap based planners, since only few simple robotic systems (e.g., the Reeds-Shepp car [35]) have analytical steering algorithms to exactly connect two states, multiple gaps in the roadmap could seriously affect the precision of the returned solutions. This might be one of the main difficulties associated with extending roadmap planners to motion planning with differential constraints.

IV. Gap Reduction Algorithms

In this section, we will review the main ideas of the gap reduction algorithms in [9]. First, the geometric properties of a class of robotic systems, which are the key to the efficiency of our algorithms, are presented. Following this, we provide the gap reduction algorithms.

Symmetries in systems: Consider a Lie group $\mathcal{G}$, with identity element e, acting on the state of the system through the (left) action $\Psi : \mathcal{G} \times \mathbf{X} \rightarrow \mathbf{X}$; we will often use the shorthand $\Psi_g(x) := \Psi(g, x)$. We call $\mathcal{G}$ a symmetry group for the system (1) if the system's dynamics are *invariant* with respect to the action of $\mathcal{G}$. Invariance is equivalent to the following statement. Given any trajectory $t \mapsto (x(t), u(t)) \in \mathbf{X} \times \mathcal{U}$ which is a solution to Equation (1), the trajectory $t \mapsto (\Psi(g, x(t)), u(t))$ is also a solution to equation 1 for all $g \in \mathcal{G}$. It can be verified that invariance implies that group actions commute with state transitions. If $x_f = \Phi^{t_f}_u(x_{\rm init})$, then $\Psi_g(x_f) = \Phi^{t_f}_u(\Psi_g(x_{\rm init}))$, i.e., $\Psi_g \circ \Phi^{t_f}_u = \Phi^{t_f}_u \circ \Psi_g$.

For many robot systems with symmetries, $\mathbf{X}$ can be partitioned, at least locally, into the Cartesian product of two manifolds $\mathbf{X} = \mathcal{G} \times \mathcal{Z}$. For simplicity of exposition, we will assume that $\mathcal{Z} = \mathbb{R}^{n_z}$, $n_z < n$. Accordingly, we write the generic point $x \in \mathbf{X}$ as the pair $(g, z) \in \mathcal{G} \times \mathcal{Z}$. Following conventions from differential geometry, $\mathcal{Z}$ is the base space, $\mathcal{G}$ is the fiber, and their product $\mathbf{X}$ is a principal fiber bundle; see [21].

Gap reduction algorithms: Consider a policy $u : [0, t_u) \rightarrow \mathcal{U}$ obtained from a discrete policy $\tilde{\pi}$ returned from an ISA. Assume that there is a gap between $\Phi^{t_u}_u(x_{\rm init})$ and x_g, a gap elimination algorithm aims at perturbing u into a new policy $v : [0, t_v) \rightarrow \mathcal{U}$ such that $\mathcal{D}(\Phi^t_v(x_{\rm init})) \leq 0$ for all $t \in [0, t_v)$ and $\rho(\Phi^{t_v}_v(x_{\rm init}), x_g) \ll \rho(\Phi^{t_u}_u(x_{\rm init}), x_g)$. In principle, the gap reduction problem can be formulated and solved as nonlinear programming problem with objective function $\rho(\Phi^{t_u}_u(x_{\rm init}), x_g)$. Such an approach is however unpractical for nonlinear systems, since the number of degrees of freedom in the choice of the control perturbation v is very large, and computing the effect of a perturbation applied at time $t \in [0, t_f)$ requires an *ex-novo* integration of the ODE (1) over $[t, t_f)$. In other words, since the final state $x_f = \Phi_u^{t_u}(x_{\text{init}}) = \Phi_{\tilde{\pi}_{k-1}}^{\Delta t} \circ \cdots \circ \Phi_{\tilde{\pi}_1}^{\Delta t} \circ \Phi_{\tilde{\pi}_0}^{\Delta t}(x_{\text{init}})$ and each state transition generally corresponds to an expensive numerical integration, perturbing u_i normally requires to recalculate state transitions for $\{\tilde{\pi}_j\}_{j=i,\cdots,k-1}$ which considerably affects the running time of the optimization.

The idea behind our approach to gap elimination is the following. Suppose that it is possible to perturb the control input $\tilde{\pi}_i$ into $v : [0, t_v) \rightarrow \mathcal{U}$, in such a way that $\Phi_v^{t_v} = \Psi_g \circ \Phi_{\tilde{\pi}_i}^{\Delta t}$ for some $g \in \mathcal{G}$. Then, since the group action Ψ and the state flow Φ commute, the new final state $x'_f = \Psi_g \circ x_f$, which intuitively means that the remaining part of the trajectory can be "rigidly translated," without need for re-integration. If it is possible to find a class of perturbations on the control input that satisfy the stated condition, then the final state of the trajectory can be written as an algebraic function of the perturbation parameters—without need for numerical integration.

In [9], we identified gap-reduction techniques applicable to systems with feedback-linearizable base dynamics, and with affine in control based dynamics. In this paper, we will consider only one technique, which according to simulation experiments, reported in the same paper, yielded by far the best results. Consider, for simplicity, a system on a principal fiber bundle, whose dynamics are expressed as $\dot{g}(t) = g(t)\xi(z(t))$, and $\dot{z}(t) = f_z(z) + g_z(z)u$. If at time $\bar{t}$, there exists $\bar{u} \in \mathcal{U}$ such that

$$f_z(z(\bar{t})) + g_z(z(\bar{t}))\bar{u} = 0, \tag{2}$$

then a trajectory segment can be inserted after $\bar{t}$, over which the base variables remain constant, and the fiber variable evolves according to $\dot{g}(t) = g(t)\xi(z(\bar{t}))$. Given any $t \in (\bar{t}, t_f)$, the generic point in the original trajectory can be written as

$$\Phi_u^t(x_0) = \Phi_{u_2}^{t-\bar{t}} \circ \Phi_{u_1}^{\bar{t}}(x_0),$$

where u_1 and u_2 are the restrictions of the control input history u to $[0, \bar{t})$, and $[\bar{t}, t_f)$, respectively. Define the perturbed control input $v_\tau : [0, t_f + \tau) \rightarrow \mathcal{U}$ as follows:

$$v_\tau(t) = \begin{cases} u(t), \text{ if } t < \bar{t}; \\ \bar{u}, \text{ if } t < \bar{t} + \tau; \\ u(t - \tau), \text{ if } t > \bar{t} + \tau. \end{cases}$$

Then, it can be verified that

$$\Phi_{v_\tau}^{t+\tau}(x_0) = \Phi_{u_2}^{t-\bar{t}} \circ \Psi_h \circ \Phi_{u_1}^{\bar{t}}(x_0) = \Psi_h \circ \Phi_u^t(x_0), \tag{3}$$

with $h = \exp(\eta\tau)$, and $\eta = g(\bar{t})\xi(z(\bar{t}))g^{-1}(\bar{t})$, i.e., the perturbed final state can be obtained by applying a certain group action to the unperturbed final state, corresponding to a flow along the vector field η for time $\tau \geq 0$.

The same argument can be repeated for a finite number of times $\bar{t}_1, \ldots, \bar{t}_k$ for which (2) holds; correspondingly, (3) can be written as

$$\Phi_{v_\tau}^{t+\tau}(x_0) = \Psi_{h_k} \circ \ldots \circ \Psi_{h_1} \circ \Phi_u^t(x_0).$$

The significance of the above equation is that it shows that the perturbed final state can be obtained from the unperturbed final state, through the combination of (positive) flows along the vector fields η_i. An immediate consequence of this is that, if the set $\{g \in \mathcal{G} : g = \prod_i \exp(\eta_i \tau_i), \tau_i \geq 0 \forall i\}$ is equal to $\mathcal{G}$, then gaps can be eliminated completely (ignoring obstacles). A catalog of vector fields that satisfy this conditions in cases of interest, together with formulas for inversion, is presented in [32].

V. Improving Planners by Gap Reduction

In this section, how to use the gap reduction algorithm [9] to improve sampling based planners will be presented. As we mentioned in Section III, obtaining high precision solutions normally requires a small gap tolerance. According to our experimental results in Fig. 4, the computational cost increases dramatically with the gap tolerance decreasing. Therefore, the key idea for the improvement is to plan low precision solutions with big gaps, and then refine solution quality by reducing the gaps. If the gaps in a low precision solutions are reduced successfully, a high precision solution is obtained. Since it is relatively easy to find a low precision solution and the gap reduction algorithm is very efficient, a high precision solution could be returned much faster.

Gaps induced by the gap tolerance in a low precision solution will be reduced in two steps since in the motion planning process these gaps exist in both the fiber and base space. The gaps in the base space are reduced first because the gap reduction algorithm in [9] reduces the gaps in the fiber space while maintaining the base variables invariant. Then after the gaps in the fiber space are reduced in the second step, gaps in both fiber and base space are reduced. To simplify the first reduction step, we only consider the base dynamics while ignoring the fiber dynamics. Since only systems with affine in control base dynamics are considered in the paper, the base gap reduction is actually a control design problem for systems affine in control. Classical techniques for systems affine in control [11], [18], [22] could be used to reduce the gaps in the base space and are not discussed here for the sake of conciseness.

Modification to ISAs: The modification to the case of the bi-directional tree based ISA is as follows and it could be easily extended to other cases.

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two trees generated from the initial and goal state, respectively. We can assume that in one iteration, $\mathcal{T}_1$ generates a new node n_{new} and $\rho(x(n_{new}), x(n)) > \epsilon_s$ for any n in $\mathcal{T}_2$. Normally, it will take much longer time for $\mathcal{T}_1$ to generate n'_{new} such that $\rho(x(n'_{new}), x(n)) < \epsilon_s$ for some n in $\mathcal{T}_2$. Now for each node n in tree $\mathcal{T}_2$, we reduce the gap between $x(n_{new})$ and $x(n)$ in two steps shown above. If the final gap is less than ϵ_s, then we find a solution. Note that since the fiber dynamics is ignored in the first step, the gaps in the fiber space could be either larger or smaller. We check the gap in fiber space after the first reduction. If it is less than an intermediate tolerance ϵ_b, which is normally much larger than ϵ_s, then the gap reduction algorithm [9] is called to reduce the gap in the fiber space.

Modification to roadmap based planners: When analytical steering methods are not available, ISAs could be used as local planners for roadmap based planners [1]. Then construction and query time of roadmap based planners will increases dramatically with the decreasing gap tolerance since both construction and query depend on the local planner. Therefore, a straightforward method is to just improve the local planner with the gap reduction algorithm. If it successfully reduces the gap to within some tolerance, then the roadmap is updated. In the roadmap construction phase, this may allow two milestones to be connected. In the query phase, same approach may be used to connect the initial and goal queries to the solution. Finally, when a solution is produced, the gap reduction may be further applied.

VI. System Models Used in Simulations

Our simulation includes three systems with affine-in-control base dynamics, which are a car with dynamics, a car-trailer system, and an underactuated spacecraft.

Car with dynamics: The state vector, in coordinates, is $(v_y, \omega, x, y, \theta)$, in which $x \in [0, 800]$, $y \in [-800, -450]$, and $\theta \in [-\pi, \pi]$ represent position and orientation; $\omega \in [-10, 10]$ is the angular velocity, $v_y \in [-10, 10]$ are translational velocity along the y-axis of the local frame fixed on the car. The input to the system is the steering angle, $u \in [-0.6, 0.6]$. The equations of motion are $\dot{v}_y = -v_x\omega + (f_{yf}(u) + f_{yr}(u))/M$, $\dot{\omega} = (f_{yf}(u)a - f_{yr}(u)b)/I$, $\dot{x} = v_x \cos(\theta) - v_y \sin(\theta)$, $\dot{y} = v_x \sin(\theta) + v_y \cos(\theta)$, $\dot{\theta} = \omega$, in which $v_x = 88$ is constant translational velocity along x-axis of the local frame, M and I are the mass and inertia, and $f_{yf}(u)$ and $f_{yr}(u)$ are affine in u and represent forces acting on front and rear tires along the y-axis of the local frame. The fiber of the system is (x, y, θ), and the base is (v_y, ω).

The car-trailer system: The car-trailer system consists of a car pulling a trailer. The state of the system can be represented using a local chart, as $(x, y, \theta_1, v, \theta_2)$, in which x, y, and θ_1 are x-, y-coordinates and orientation of the car, v is the translational velocity along x-axis of the local frame, and θ_2 is the orientation of the trailer. The input to the system is (u_1, u_2), in which u_1 is the changing rate of v, and u_2 controls the steering angle. The motion equations of the trailer system are as follows: $\dot{x} = v\cos(\theta_1)$, $\dot{y} = v\sin(\theta_1)$, $\dot{\theta}_1 = vu_2/L_1$, $\dot{v} = u_1$, $\dot{\theta}_2 = v\sin(\theta_1 - \theta_2)/L_2$, in which L_1 is the length of the car, L_2 the length of the hitch. The system has affine-in-control base dynamics could be seen by introducing the transformation $\theta_d = \theta_1 - \theta_2$ and change the last equation above to $\dot{\theta}_d = vu_2/L_1 - v\sin(\theta_d)/L_2$. The fiber of the system is (x, y, θ_1) and the base is (v, θ_d). The state space bounds are: $x \in [0, 400]$, $y \in [0, 400]$, $\theta_1 \in [-\pi, \pi]$, $v \in [0, 5]$, and $\theta_2 \in [-\pi, \pi]$. The bounds on inputs are $u_1 \in [-1, 1]$ and $u_2 \in [-0.8, 0.8]$. Note, we intentionally set v, the forward speed, to be nonnegative so that the system is not STLC, and the gap cannot be reduced by trivially moving along the direction of Lie bracket.

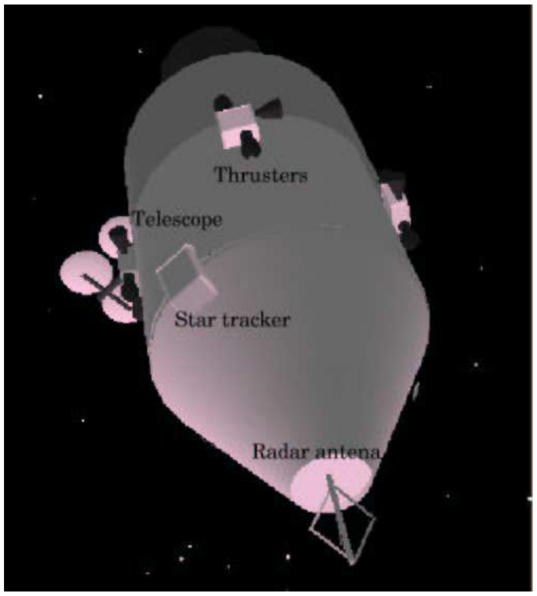

Fig. 3. A spacecraft equipped with a star tracker, telescope and a communication antenna. Note that this model is a fictional model.

The spacecraft with orientation constraints: This problem is from [16] with some modifications. Dynamics of both translation and rotation of the spacecraft is considered. Also, we assume that some of the thrusters on the spacecraft (Fig. 3) do not work because of malfunction. The translational forces and rotational torques for the spacecraft will only be provided from three pairs of thrusters. Each pair could provide forces in opposite, independent directions, and these forces do not pass through the mass center; the system is hence underactuated but controllable. The state of the system is represented by (g, ξ), in which $g = (p, R) \in SE(3)$ represents position, $p \in \mathbb{R}^3$ and orientation, $R \in SO(3)$, and $\xi = (\hat{\Omega}, V) \in se(3)$ denote the translational and rotational velocity expressed in the body frame. The skew matrix $\hat{\Omega}$ is defined as unique matrix for which $\hat{\Omega}v = \Omega \times v, \forall v \in \mathbb{R}^3$. The motion equations are: $\dot{R} = R\hat{\Omega}$, $\dot{p} = RV$, $\dot{\Omega} = -J^{-1}(\Omega \times (J\Omega) - f_\Omega)$, $\dot{V} = V \times \Omega + f_V/M$, in which J is the inertial matrix, M is the mass, $f_V = [u_0, u_1, u_2]^T$ is the translational force vector, $f_\Omega = [-u_2L_z, u_0L_x, u_1L_y]^T$ is the rotational torque vector and L_x, L_y, and L_z are vertical distance from the mass center to the direction of forces generated by thrusters. The input to the system is u_0, u_1 and u_2.

Besides dynamics constraints and the space station in the environment, constraints also come from the star tracker, the telescope and the communication antenna on the spacecraft. Specifically, the antenna should not be too

Model	Current gap tolerance / Smallest gap tolerance			
	1	10	20	30
1	78560.2	11265.2	3881.8	2647.4
2	143527.6	22395.3	420.9	193.92
3	570080.0	11044.4	6854.5	6735.2

Fig. 4. Total running times in seconds for 20 trials to use the old planner to solve same problems with different gap tolerance, in which "1", "2" and "3" denote the problem with the car, the trailer, and the spacecraft system, respectively.

far away from the direction of the Earth to maintain the communication. To protect sensitive equipments, the star tracker cannot be close to the direction of the Sun, and the telescope cannot be close to the direction of any bright objects (such as the Sun, Moon, Jupiter and Earth). In our implementation, we assume that the antenna is at the top of the spacecraft along the local z axis. The direction of telescope and star tracker are along the local x and y axis, respectively. In our implementation, sitting at the origin of the inertial frame, the light from bright objects comes in parallel from different directions. Specifically, we assume that the light of the Earth is from the direction of $(0, 0, 1)$, the light of the Sun is from the direction of $(1, 1, 1)$, the light of the Jupiter is from the direction of $(1, 1, 0)$, and the light of the Moon is from the direction of $(1, 0, 1)$.

VII. Simulation Studies

We improve the performance of two sampling-based planners by gap reduction. One is a bidirectional ISA based on Resolution Complete Rapidly-Exploring Random Trees (RC-RRTs) [10], and the other is a basic PRM-based planner. The simulation is done on a 2.0 Ghz PC running Linux. The NAG library is used to solve nonlinear programs in the gap reduction technique.

A motion planning problem is constructed for each system from Section VI. The problems and sample solutions are shown in Figs. 5, 6 and 1, respectively.

The first set of simulations is used to show the relationship between the running time and the gap tolerance. An old RC-RRT based planner solves the same problems with gap tolerance of different sizes. For each gap tolerance, we run the old planner 20 times and report the total running time. The smallest gap tolerance for problems with the car, the trailer, and the spacecraft is 0.37, 0.1, and 3.0, respectively. The results are shown in Fig. 4, from which we can see that the running time of the old planner increases dramatically with the gap tolerance decreasing.

In the second set of simulations, we compare the running time to find solutions with similar gap tolerance. The search tolerance ϵ_s for all problems is around 0.1. Both the old planner and the improved planner are used to solve same problems 40 times. The average, maximum and minimum running time are reported.

The comparison of running time is given in Table 7. From the table, it is easy to see that the running time to find high precision solutions is considerably reduced.

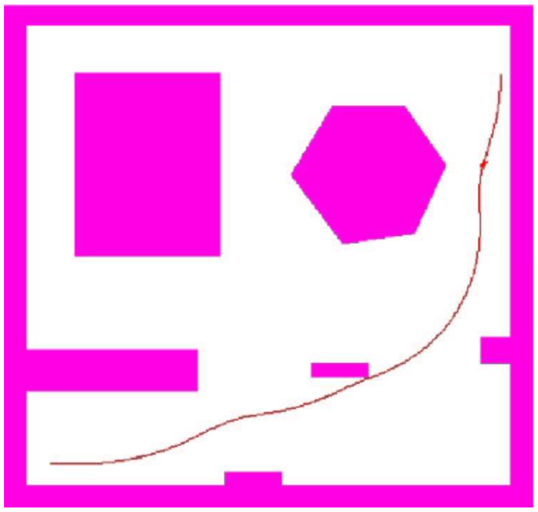

Fig. 5. Trajectory design for the car with dynamics.

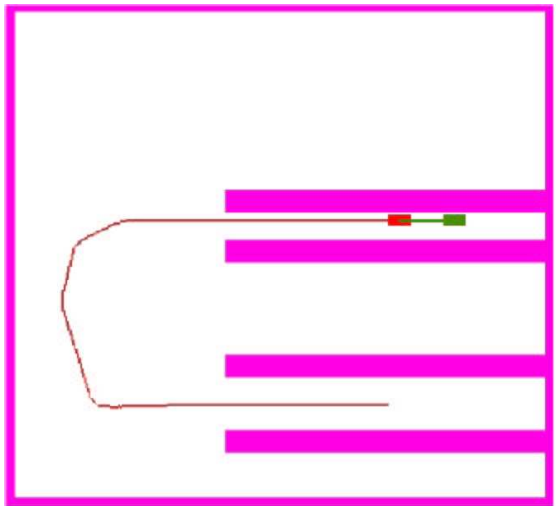

Fig. 6. Trajectory design for the trailer system.

Finally, the PRM planner is test on a problem with a unicycle car with the acceleration control. The car model is $\dot{x} = v\cos(\theta), \dot{y} = v\sin(\theta), \dot{\theta} = vu_2/L_1$ and $\dot{v} = u_1$, in which (x, y, θ) are the position and orientation, v is the forward speed (nonnegative in our simulation) in the local frame, L_1 is the length of the car, u_1, u_2 control the acceleration and steering angle, respectively. The above improved ISA is used as the local planner with the gap tolerance 0.1. In the construction stage, a directed edge is added between two states (including the velocity) when the local planner successfully connects them. In the multiquery stage, if both initial and goal states could be connected to the roadmap, a solution is returned. After building the roadmap for each 25 iterations, we will query solutions for 50 randomly chosen initial and goal state pairs. The results are shown in Fig. 9 and a partial roadmap is in Fig. 8.

Model	Improved Planner			Old Planner		
	Av. (s.)	Max. (s.)	Min. (s.)	Av. (s.)	Max. (s.)	Min. (s.)
1	160	550	62	24256	69714	247
2	96	303	2	7176	20285	436
3	5562	33883	504	N/A	N/A	N/A

Fig. 7. Comparison of two planners on running time on different motion planning problems, in which "1", "2" and "3" represent the car, trailer, and spacecraft system, "Av.", "Max." and "Min." represent the average, maximum, and minimum running time over all trials, "N/A" means that no solution is found after about 72 hours.

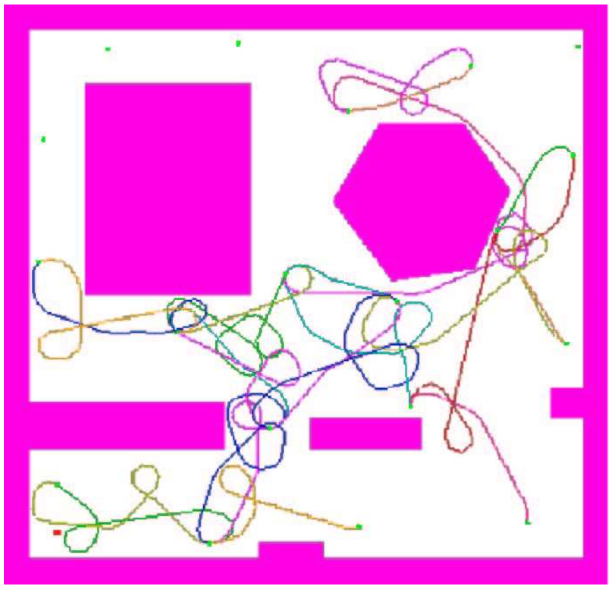

Fig. 8. A partial roadmap built by the PRM-based nonholonomic planner.

VIII. Conclusion

This paper establishes the power of efficient gap reduction techniques in sampling-based motion planning under differential constraints. New planners are designed by combining the improved gap reduction algorithm [9] with an RC-RRT based planner in [10] and a simple PRM based planner. The comparison of RC-RRT based planners with or without gap reduction on the running time to solve same problems demonstrated dramatic performance improvements. We expect that techniques such as this will enable many new challenging problems to be solved. Furthermore, the performance of the primitive PRM-based planner showed how the gap reduction techniques can be used to construct roadmaps without requiring an accurate steering method. However, substantial work remains to make this approach to PRMs viable for applications.

Acknowledgments

This work was funded in part by the following grants: NSF CAREER 9875304 (LaValle), NSF 0208891 (Frazzoli and LaValle), and NSF 0118146 (Bullo and LaValle).

Iteration	C. T.(s)	Q. T.(s)	Success	N.N.	E.N.
25	378.73	210.99	3/50	13	19
50	2121.69	1161.22	21/50	38	118
75	2084.97	1819.72	25/50	61	213
100	1589.97	2453.82	25/50	85	309
125	1731.32	2365.04	24/50	107	398
150	1911.5	2776.65	34/50	130	501
175	1863.71	2646.88	36/50	151	607

Fig. 9. Simulation results for the PRM based nonholonomic planner, in which "C.T." is the construction time, "Q.T." is the overall query time for 50 trials, "N.N." is the number of milestones, and "E.N." is the number of edges in the roadmap.

IX. References

[1] M. Akinc, K. Bekris, B. Chen, A. Ladd, E. Plakue, and L. Kavraki. Probabilistic roadmaps of trees for parallel computation of multiple query roadmaps. In *Eleventh International Symposium on Robotics Research*, 2003.

[2] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. In *IEEE Int. Conf. Robot. & Autom.*, pages 2328–2335, 1991.

[3] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[4] J. Bobrow, S. Dubowsky, and J. Gibson. On the optimal control of robotic manipulators with actuator constraints, 1983.

[5] F. Bullo and K. M. Lynch. Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems. *IEEE Trans. on Robotics and Automation*, 17(4):402–412, 2001.

[6] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. *Discrete and Computational Geometry*, 6:461–484, 1991.

[7] J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[8] J. F. Canny, B. Donald, J. Reif, and P. Xavier. The complexity of kinodynamic planning. In 29^{th} *Symposium on the Foundations of Compute Science*, 1989.

[9] P. Cheng, E. Frazzoli, and S. LaValle. Exploiting group symmetries to improve precision in kinodynamic and nonholonomic planning. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2003.

[10] P. Cheng and S. LaValle. Resolution complete rapidly-exploring random trees. In *IEEE Int. Conf. Robot. & Autom.*, 2001.

[11] J. Coron, L. Praly, and A. Teel. Feedback stablilization of nonlinear systems: sufficient conditions and lyapunov and input-output techniques. In A. Isidori,

editor, *Trends in Control: A European Perspective*, pages 293–348. Springer, London, 1995.

[12] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14(6):443–479, 1995.

[13] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic planning. *Journal of the ACM*, 40:1048–66, November 1993.

[14] E. Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, June 2001.

[15] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[16] E. Frazzoli, M. A. Dahleh, E. Feron, and R. Kornfeld. A randomized attitude slew planning algorithm for autonomous spacecraft. In *AIAA Guidance, Navigation, and Control Conference*, 2001.

[17] J. Hollerbach. Dynamic scaling of manipulator trajectories. Technical report, MIT A.I. Lab Memo 700, 1983.

[18] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, Berlin, 1989.

[19] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, June 1996.

[20] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. In *IEEE Int. Conf. Robot. & Autom.*, 2000.

[21] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry. Vol. I*, volume 15 of *Interscience Tracts in Pure and Applied Mathematics*. Interscience Publishers, New York, NY, 1963.

[22] M. Krstic and H. Deng. *Stabilization of Nonlinear Uncertain Systems*. Springer-Verlag, Berlin, 1998.

[23] G. Laffierriere and H. J. Sussman. *Nonholonomic Motion Planning*, chapter A Differential Geometric Approach to Motion Planning. The Kluwer International Series in Engineering and Comuter Science, Boston, MA, 1992.

[24] F. Lamiraux and D. Bonnafous. Reactive trajectory deformation for nonholonomic systems: Application to mobile robots. In *IEEE Int. Conf. Robot. & Autom.*, pages 3099–3104, 2002.

[25] J. P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In J.-P. Laumond, editor, *Robot Motion Plannning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[26] S. M. LaValle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *International Journal of Robotics Research*, 20(9):729–752, September 2001.

[27] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.

[28] J. Luh and C. S. Lin. Optimum path planning for mechanical manipulators. *J. Dyn. Sys. Meas. Contr.*, 102:142–151, 1981.

[29] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. *Int. J. Robot. Res.*, 15(6):533–556, 1996.

[30] A. Marigo and A. Bicchi. Steering driftless nonholonomic systems by control quanta. In *IEEE Conf. Decision & Control*, 1998.

[31] A. Marigo, B. Piccoli, and A. Bicchi. Reachability analysis for a class of quantized control systems. In *Proc. IEEE Conf. on Decision and Control*, 2000.

[32] S. Martinez, J. Cortes, and F. Bullo. A catalog of inverse-kinematics planners for underactuated systems on matrix lie groups. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 625–630, 2003.

[33] R. M. Murray and S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *Trans. Automatic Control*, 38(5):700–716, 1993.

[34] S. Pancanti, L. Leonardi, L. Pallottino, and A. Bicchi. Optimal control of quantized input systems. In C. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control V*, Lecture Notes in Computer Science. Springer, 2002.

[35] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific J. Math.*, 145(2):367–393, 1990.

[36] J. Reif and H. Wang. Non-uniform discretization approximations for kinodynamic motion planning. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 97–112. A K Peters, Wellesley, MA, 1997.

[37] P. Rouchon, M. Fliess, M. Levine, and P. Martin. Flatness, motion planning, and trailer systems. In *Proc. IEEE Conf. on Decsion and Control*, pages 2700–2705, 1993.

[38] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *Int. J. Robot. Res.*, 17:840–857, 1998.

[39] K. G. Shin and N. D. McKay. Minimum-time control of robot manipulators with geometric path constraints. *IEEE Trans. Autom. Control*, 30(6):531–541, 1985.

[40] P. Svestka and M. H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *IEEE Int. Conf. Robot. & Autom.*, pages 1631–1636, 1995.