Improving the Performance of Sampling-Based Motion Planning with Symmetry-Based Gap Reduction

Peng Cheng, Emilio Frazzoli, and Steven LaValle

***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 paths due to various factors. These gaps often cause that higher precision solutions come at the expense of dramatically increased computation time. In this paper, we propose a gap reduction technique that substantially improves the performance of sampling-based algorithms by perturbing the trajectory and eliminating large gaps in solution path candidates. By exploiting group symmetries of the system, gap reduction is greatly accelerated by avoiding unnecessary costly numerical integrations and naturally maintaining local constraints on the system. The improvement is demonstrated for unidirectional, bi-directional, and PRM-based sampling-based algorithms in solving problems with a nonholonomic system and a system with dynamics.**

***Index Terms*— Motion planning, nonholonomic planning, kinodynamic planning, differential constraints, symmetry**

Submitted on October 17, 2005

P. Cheng (corresponding author) is with General Robotics, Automation, Sensing, and Perception (GRASP) Laboratory, University of Pennsylvania, Philadelphia, PA 19104-6228 USA (e-mail: chpeng@seas.upenn.edu)

E. Frazzoli is with Mechanical and Aerospace Engineering Department, University of California Los Angeles, Los Angeles, CA 90095 USA (e-mail: frazzoli@ucla.edu)

S. LaValle is with Department of Computer Science, University of Illinois, Urbana, IL 61801 USA (e-mail:lavalle@cs.uiuc.edu)

I. Introduction

This paper addresses motion planning with differential constraints (MPD), in which, besides the global constraints due to obstacles in the environment, local differential constraints also exist that restrict the available velocities and accelerations of the robotic systems at each state (which includes configuration and velocity). Typical MPD problems include nonholonomic planning [1] and kinodynamic planning [2]. Without differential constraints, the motion planning problem is generally called the *Generalized Movers' Problem* [3]. By the concept of configuration space, the robotic system is represented by a point in the configuration space, and the motion planning problem is transformed into a pure geometry problem, whose solution is a continuous collision-free path connecting the initial and goal configurations in the configuration space. For MPD problems, the robotic system is modeled as a control system. The changing rates of the states of the system are determined by control inputs. For example, the acceleration of a car is determined by the gas pedal, the brake, and forces on the steering wheel. The set of all possible input values is called the input space. The task is to find a control, which is a function from a time interval to the input

space.[1] As common in the motion planning literature, we consider the case in which the system dynamics is known, and there are no uncertainties and disturbances, and an open-loop control is adequate. [2] Applying the control, the robotic system will move from the initial state to the goal state without colliding into obstacles. The resulting state history is called the trajectory of the control, which is a function from the time interval into the state space.

Most of current algorithms for MPD problems use sampling-based techniques [2], [5]–[14], in which a search graph is incrementally built to construct solutions using a set of controls sampled from the control space, which is a set that includes all possible controls. The search graph is a directed graph, for which nodes are associated with states, and edges are associated with sampled controls and trajectories. The control of a path in the search graph is constructed by sequentially concatenating the controls of the edges in the path. The *concatenation* of two controls means to append one control to the end of the other one by offsetting the time interval of one control by the duration of the other control. Initially, the search graph could include a finite number of nodes. In each iteration, a node $n_{\rm cur}$ in the search graph is firstly chosen by a global search strategy. A local planner uses a sampled control $\tilde{u}_{\rm new}$ to extend the state of $n_{\rm cur}$ to a new state while considering differential constraints. The search graph is updated with a specified policy. If there exists a *solution path*, which satisfies a given solution checking criterion, the control of the solution path is returned as a solution; otherwise,

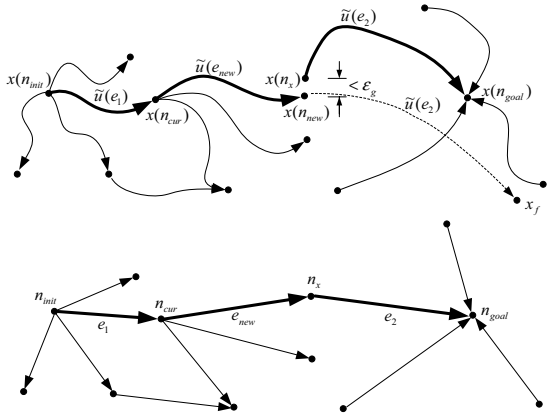

Fig. 1. An example of the search graph and solution path.

the algorithm iterates until a given termination condition is satisfied. An example of the search graph and solution path is in Fig. 1, in which nodes and edges in the search graph in the lower picture are denoted respectively as n and e, the state and control associated with node n and edge e in the search graph are respectively denoted $x(n)$ and $\tilde{u}(e)$ in the upper picture, and the solid curve below $\tilde{u}(e)$ is the trajectory associated with edge e. The thick lines in the search graph show the solution path, the thick lines in the upper picture are trajectories of the edges of the solution path, and the solution is the control of the solution path that is the concatenation of $\tilde{u}(e_1), \tilde{u}(e_{\rm new})$, and $\tilde{u}(e_2)$.

Even though sampling-based planning algorithms have solved many challenging problems, one common problem is that their solution path could have gaps between: 1) the final state of the trajectory of an edge and the starting state of the trajectory of the next edge in the path, 2) the state of the starting node of the path and the initial state, or 3) the state of the final node of the path and the goal state. An example of the gap from bi-directional search is shown in Fig. 1. For a bi-directional search method, the search graph initially

[1]As in [4], "input" is distinguished from "control" in this paper. "Input" is a value in the input space, and "control" is a function from a time interval to the input space.

[2]In realistic applications, some form of feedback will be needed, but this would be out of the scope of the paper.

consists of two disjointed subgraphs, which start from nodes $n_{\rm init}$ and $n_{\rm goal}$, respectively. One expands forward in time, and the other one expands backward in time. If the distance between a new state, $x(n_{\rm new})$, in one subgraph and a state $x(n_x)$ in the other subgraph is less than a given gap tolerance ϵ_g, two subgraphs are connected by unifying these two nodes as one node. In the case shown in Fig. 1, the unified node is n_x. The solution path is from the initial state node, $n_{\rm init}$, to the goal state node, $n_{\rm goal}$. Similarly, the search graph of PRM-based planners is initialized with multiple disjoint subgraphs. Two subgraphs are connected with a unifying node when the distance between the states of two nodes from two subgraphs is less than a given gap tolerance. If a solution path exists and traverses multiple subgraphs, it could have a gap for each unified node along the path.

Gaps greatly degrade the quality of the solutions, especially for bi-directional or PRM-based search. As shown in the upper picture of Fig. 1, when a small gap changes the state from $x(n_x)$ to $x(n_{\rm new})$, the final state may change greatly, from $x_{\rm goal}$ to $x_{\rm f}$, after integration over the same control $\tilde{u}(e_2)$. Therefore, planners are expected to use small gap tolerances. However, under control-space sampling, some systems are not small-time locally controllable (STLC): the sets of reachable states from the initial state have the structure of a lattice, which prevents the exact matching of the trajectory endpoint with the goal state. A solution path may not be found if the smallest distance between the reachable states and the goal state is larger than the given gap tolerance. In cases in which the set of reachable states is everywhere dense [15], or even continuous [16], [17], it is possible in principle to add a sequence of sampled controls to move the final state arbitrarily close to the goal state, but this is done at the expense of the efficiency of the trajectory and longer running time since there will be fewer solution paths, and the search depth, i.e., the number of edges, of solution paths tends to increase.

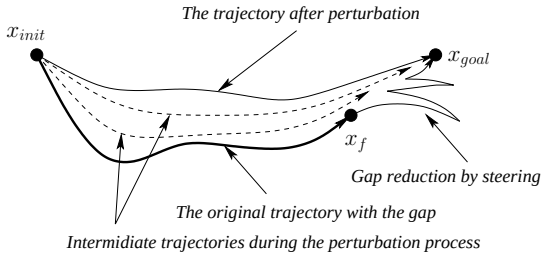

Fig. 2. Comparison of the gap reduction by perturbation and steering.

Note that a planner can often quickly return solution paths with a large gap tolerance because the solution paths have low search depth. If a separate algorithm could efficiently reduce the gaps, then planners can quickly find a solution by first finding solution path candidates that have large gaps and then reducing their gaps. One way to reduce gaps is to use analytical solutions for steering problems to design a trajectory that connects two end states of the gap. However, there are only few analytical solutions [18]–[23]. Furthermore, the cost of the resulting trajectories might be dramatically increased at the gaps. For example, for a kinematically controllable system [19], the system stops when switching between decoupling vector fields during the steering process. For example, if a vehicle has a high speed along a decoupling vector field at one gap endpoint and a vector field switch is necessary to eliminate the gap, stopping the system and switching to another decoupling vector field could take a long time.

We developed a symmetry-based gap reduction method [24], [25] that reduces the gaps by perturbing the solution path candidate. The difference between gap reduction by perturbation and steering is illustrated in Fig. 2. In this paper, the gap reduction problem is cast as an optimization of the distance between two

gap endpoints. The distance is efficiently evaluated by using group symmetries of systems to avoid expensive numerical integrations. Another advantage of our method is that many local constraints on states and inputs can be naturally enforced in the trajectory after symmetry-based perturbations. Besides our work, a similar method is presented in [26] by tailoring the reactive path deformation method [27]. In this paper, we present gap reduction algorithms that use symmetries, and combine them with different types of sampling-based planners, which include the unidirectional and bi-directional RRT (Rapidly-exploring Random Tree)-based planner [7] and a PRM-based planner [28]. By comparing results from problems with both a nonholonomic system and a system with dynamics, the new planners improved by our gap reduction algorithms dramatically outperform the original ones.

II. Sampling-Based MPD and Gap Problems

This section formally defines MPD problems, and presents a template for general sampling-based algorithms, from which we describe how the gap problem is generated and solved.

A. MPD problems

An MPD problem is a tuple, $(X, U, \mathcal{U}, X_{\rm obs}, f, x_{\rm init}, x_{\rm goal})$. The *state space*, X, is a differentiable manifold of dimension n. Without losing generality, the *input space*, $U \subset \mathbb{R}^m$, is assumed to be

$$U = [-1, 1]^m \tag{1}$$

in which $m \leq n$. An element of U is called an *input* and is denoted as u. Keep in mind that in this paper an input represents a value in $\mathbb{R}^m$, and a *control*, denoted as $\tilde{u}$, represents a piecewise-continuous vector-valued function from $[0, t]$ to U for some $t > 0$. The *control space*, $\mathcal{U}$, contains all possible controls for the system. The concatenation operator, denoted as $\circ$, of two controls $\tilde{u}_1$ and $\tilde{u}_2$ in $\mathcal{U}$ is defined as

$$(\tilde{u}_1 \circ \tilde{u}_2)(t) = \begin{cases} \tilde{u}_1(t) & t \in [0, \bar{t}(\tilde{u}_1)) \\ \tilde{u}_2(t - t_1) & t \in [\bar{t}(\tilde{u}_1), \bar{t}(\tilde{u}_1) + \bar{t}(\tilde{u}_2)], \end{cases} \tag{2}$$

in which $t > 0$, and $\bar{t} : \mathcal{U} \to (0, \infty)$ gives the time duration of control $\tilde{u} \in \mathcal{U}$.

The closed set $X_{\rm obs} \subset X$ includes the global constraints from obstacles and other inequality constraints on states. Let $X_{\rm free}$ denote the *violation-free* open set $X \backslash X_{\rm obs}$. Equality holonomic and differential constraints are encoded in a *motion equation*, f, which is a set of Ordinary Differential Equations (ODEs). We assume that these ODEs have unique solutions and are time-invariant in the following form:

$$\dot{x} = f(x, u), \quad \forall x \in X, u \in U. \tag{3}$$

The *trajectory* of a control $\tilde{u}$ from a state x_i is a state transition map

$$\Phi_{\tilde{u}} : X \times [0, \bar{t}(\tilde{u})] \to X, \tag{4}$$

which is defined as

$$\Phi_{\tilde{u}}(x_i, t) = x_i + \int_0^t f(x(\tau), \tilde{u}(\tau)) d\tau. \tag{5}$$

Note that

$$\Phi_{\tilde{u}}(x_i, 0) = x_i. \tag{6}$$

The *initial state* is $x_{\rm init}$ and the *goal state* is $x_{\rm goal}$. A control $\tilde{u}$ is an exact solution to the give problem if its trajectory $\Phi_{\tilde{u}}(x_{\rm init}, t)$ satisfies:

$$\Phi_{\tilde{u}}(x_{\rm init}, t) \in X_{\rm free}, \text{for all } t \in [0, \bar{t}(\tilde{u})] \tag{7}$$

and

$$\Phi_{\tilde{u}}(x_{\rm init}, \bar{t}(\tilde{u})) = x_{\rm goal}. \tag{8}$$

An example of an MPD problem [29] is shown in Fig. 3, in which a car with realistic dynamics [30] is

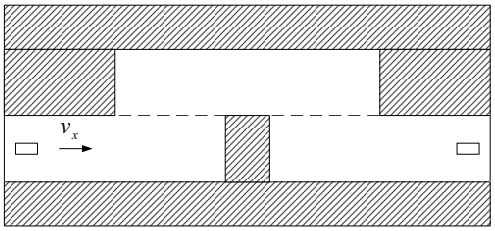

Fig. 3. Consumer Union Short Course.

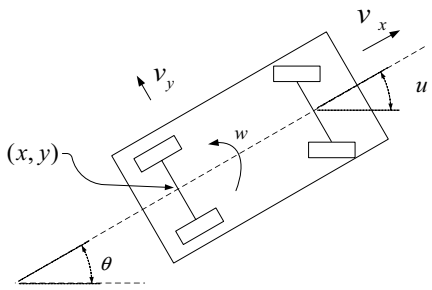

Fig. 4. The sketch of the car with dynamics.

required to do a lane changing maneuver on a two lane road with 60mph constant forward speed. The constant speed is achieved by appropriately using the brake and gas pedals. The shaded areas are obstacles to keep the car on the road and complete the lane change. This problem is referred to in the automotive industry as the Consumer Union Short Course [31].

The state of the car in Fig. 4 is $(x, y, \theta, v_y, \omega)$, in which $x \in [0, 800]$, $y \in [-800, -450]$, and $\theta \in [-\pi, \pi]$ represent position and orientation; $\omega \in [-5, 5]$ is the angular velocity; $v_y \in [-50, 50]$ is translational velocity perpendicular to the forward direction. The input is the steering angle, denoted as u. The input space is $[-0.6, 0.6]$. The control space includes all piecewise-continuous controls. The following motion equation is adapted from [30]

$$\begin{aligned} \dot{x} &= v_x \cos(\theta) - v_y \sin(\theta) \\ \dot{y} &= v_x \sin(\theta) + v_y \cos(\theta) \\ \dot{\theta} &= \omega \\ \dot{v}_y &= -v_x \omega + (f_{\rm yf} + f_{\rm yr})/M \\ \dot{\omega} &= (f_{\rm yf} a - f_{\rm yr} b)/I, \end{aligned} \tag{9}$$

in which a and b are respectively the distance from the front and rear axles to the car mass center,

$$f_{\rm yf} = -C_{\rm f}((v_y + a\omega)/v_x - u) \tag{10}$$

and

$$f_{\rm yr} = -C_{\rm r}(v_y - b\omega)/v_x \tag{11}$$

are forces acting on front and rear tires along the direction perpendicular to the forward direction, $C_{\rm f}$ and $C_{\rm r}$ are constant coefficients of $f_{\rm yf}$ and $f_{\rm yr}$, v_x is the constant forward velocity, and M and I are the mass and inertia, respectively. The parameters are provided in the appendix.

The initial and goal configuration are respectively shown as small rectangular boxes in the left and right sides of Fig. 3, and both v_y and ω are zero. The task is to design a control, which will drive the car from the initial state to the goal state without colliding into obstacles, i.e., the car completes the required lane changes.

B. A template for sampling-based MPD

Without differential constraints, sampling-based planners use sampled configurations to represent the collision-free configuration space. However, if differential constraints exist, sampling-based planners use sampled controls to represent the control space, and construct solutions with the search graph. The search graph is a directed graph, denoted by $G(N, E)$. A node $n \in N$ is associated with a state $x(n) \in X$, and a directed edge $e(n_i, n_{\rm e}) \in E$ from node n_i to $n_{\rm e}$ is associated with a

control $\tilde{u}(e)$ and a trajectory of $\tilde{u}(e)$ from state $x(n_i)$ or $x(n_e)$. [3]

Since it is normally difficult to use sampled controls to exactly connect two given states, a gap tolerance ϵ_g is usually given for checking approximate solutions and updating the search graph. [4] For a given MPD problem, and a gap tolerance, ϵ_g, a template [32] for sampling-based planning with differential constraints is as follows:

1) **Initialize Search Graph:** The search graph initially has one, two, or multiple nodes and no edges. One starting node $n_{\rm init}$ is always associated with the initial state [6] or a state in its neighborhood [2]. Other starting nodes could be associated with the goal state [11] or other states [25].
2) **Select Node:** Use a global search algorithm to choose a node $n_{\rm cur} \in N$. The global search could be Dijkstra's algorithm [6], depth-first search, breadth-first search, A^* search [33], and informed search using other heuristics [11], [34], [35].
3) **Generate Trajectory Segment:** Use a local planner to sample a control $\tilde{u}_{\rm new} \in \mathcal{U}$ and generate its trajectory from $x(n_{\rm cur})$. The final state of the trajectory is $x_{\rm new}$. The sampled controls could be constant [2], [6], [11], or kinematically decoupling vector fields [8]. The durations of these sampled controls could be a fixed value [6], [11] or a varying value [9].
4) **Update Search Graph:** Use some policy to update G. The policy usually checks first whether a new edge $e_{\rm new}$ associated with $\tilde{u}_{\rm new}$ and its trajectory should be added. If so, then one node of the edge is always $n_{\rm cur}$. For the other node, some planners [11] always introduce a new node, $n_{\rm new}$, which is associated with state $x_{\rm new}$; while some planners [7] use an existing node $n_x \in N$ if $x_{\rm new}$ is in the ϵ_g neighborhood of $x(n_x)$.
5) **Check for Solution:** Use a solution checking criterion to determine whether a solution path exists and traverses the new edge $e_{\rm new}$. If so, return the control of the path as the solution. Assume that the path consists of edges

$$\{e_1, e_2, \cdots, e_{\rm new}, \cdots, e_k\}. \tag{12}$$

The returned solution is

$$\tilde{u} = \tilde{u}(e_1) \circ \tilde{u}(e_2) \circ \cdots \circ \tilde{u}(e_{\rm new}) \circ \cdots \circ \tilde{u}(e_k). \tag{13}$$

6) **Check Termination Condition:** Go to Step 2 until a given termination condition is satisfied.

Various sampled-based algorithms could be put into the above template with differences in each step. To be more specific, we will describe the unidirectional RRT-based planner [11] in the above template. Given an MPD problem, a distance function, and a gap tolerance ϵ_g, RRT is a data structure which could be incrementally built to quickly explore the search space and construct solutions.

In Step 1, the RRT is initialized with only one node $n_{\rm init}$ whose state is $x_{\rm init}$. In Step 2, the global search algorithm chooses a node $n_{\rm near}$, whose state $x(n_{\rm near})$ is the closest to a randomly generated state $x_{\rm rand}$ with respect to the given distance function. The local planner will choose a control from a finite set of sampled controls to generate a violation-free trajectory from $x(n_{\rm near})$, whose final state $x_{\rm new}$ is the closest to $x_{\rm rand}$ of all final states of trajectories of controls in the finite set from $x(n_{\rm cur})$. All sampled controls are constant and have

[3] If the trajectory is from state $x(n_i)$ of the starting node, then the integration is forward in time; otherwise, it is backward in time.

[4] Different tolerances could be used for the solution checking and the search graph update. However, to simplify the description, we assume that they are the same here since we can always choose the smaller one for both of them.

fixed durations. For example, a finite set of sampled controls for the problem in Fig. 3 could include three controls, which have the steering angle at $-0.3, 0.0$, and 0.3 for 1 second, respectively. The update policy is to add a new node $n_{\rm new}$ for each new state $x_{\rm new}$ and a new edge $e(n_{\rm near}, n_{\rm new})$ to $\mathcal{T}$. The solution checking criterion is whether the state of the new node $n_{\rm new}$ is in the ϵ_g neighborhood of $x_{\rm goal}$. If yes, then the control of the path from $n_{\rm init}$ to $n_{\rm new}$ is returned as a solution. The termination condition is whether a given number of iterations is reached.

C. Gap problems in sampling-based algorithms

Gaps are induced in a solution path if 1) the sampled control of one of its edges does not exactly connect two states of the end nodes of the edge; 2) the starting node of the path is not associated with $x_{\rm init}$; or 3) the final node of the path is not associated with $x_{\rm goal}$. Gaps could be induced in the following steps of the algorithm template:

- **Step 1:** Gaps could be induced if there are no starting nodes associated with the initial state $x_{\rm init}$ or the goal state $x_{\rm goal}$.
- **Step 4:** A gap could be induced in a new edge if the control of the edge does not exactly connect states of two nodes of the edge. An example is shown in Fig. 5, in which the end node of the new edge is an existing node n_x in N, whereas $x_{\rm new}$ is only in ϵ_g neighborhood of $x(n_x)$. The gap is between the final state, $x_{\rm new}$, of trajectory of $e_{\rm new}$ and state $x(n_x)$
- **Step 5:** A gap is induced if the final node of the solution path is not associated with $x_{\rm goal}$.

Gaps can seriously degrade the quality of a solution, especially for bi-directional search and PRM-based search. Therefore, we are often required to use small ϵ_g in sampling-based algorithms. However, when the gap tolerance ϵ_g decreases, the time to find a solution normally increases dramatically. For example, a unidirectional planner uses breadth-first search as the global search strategy. If the state of a new node lies in the ϵ_g neighborhood of $x_{\rm goal}$, then a solution is returned. As ϵ_g decreases, the number of reachable states from $x_{\rm init}$ in the ϵ_g neighborhood of $x_{\rm goal}$ will decrease, and the search depth of these reachable states will be higher, which means a longer running time to return a solution. In the worst case, if the reachable states with a set of sampled controls form a lattice structure, and the smallest distance between $x_{\rm goal}$ and all reachable states is larger than ϵ_g, then the sampling-based algorithm cannot return a solution even though a solution does exist for the given problem.

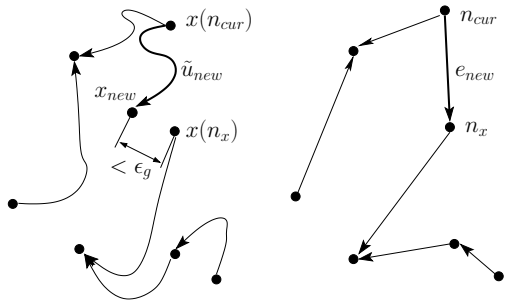

Fig. 5. An example of a gap induced in updating the search graph.

III. Motion Planning with Gap Reduction

Our approach to the gap problems is to efficiently eliminate large gaps in solution path candidates by perturbation. Noticing that the sampled controls in the candidates are only a small subset of the original control space $\mathcal{U}$, we could perturb these sampled controls in $\mathcal{U}$ to eliminate the gaps. With the gap reduction algorithms, even though the final state of a path is outside the ϵ_g neighborhood of the goal state, it could be transformed into a solution path. Therefore, a solution could be

returned sooner since the number of solution paths is increased, and these paths have lower search depth. In the following section, we will first describe the gap reduction algorithms and then incorporate them with sampling-based algorithms.

A. Gap reduction by perturbation

To be concise, we will consider a solution path candidate that only has one gap between the state of its final node and the goal state. Other types of gaps could be eliminated with minor modifications.

Assume that the control of the path candidate is $\tilde{u}$, the objective of the gap reduction algorithm is to perturb $\tilde{u}$ into $\tilde{u}'$ such that the distance between the goal state and the final state of the trajectory of $\tilde{u}'$ from the initial state is less than a given tolerance, and the new perturbed trajectory satisfies all constraints from the MPD problem. In this paper, the gap reduction problem is cast as an optimization problem to minimize the distance between the final state and goal state by perturbating the parameters, which fully determine the final state.

For a control $\tilde{u}$, a given MPD problem, and a distance function,[5] the outline of general gap reduction algorithms is:

1) **Parameterize Final State Calculation:** Determine a set of parameters, which determine the final state of the trajectory of the control under perturbation.
2) **Select a Subspace for Optimization:** Select a subset of parameters for the optimization. If the number of parameters for the final state is more than the number of parameters necessary to eliminate the gap, then a subset of parameters could be chosen to avoid the expensive evaluation of high-dimensional gradient vectors in the optimization.
3) **Optimize in the Subspace:** Perturb the selected parameters to minimize the distance between the final state and the goal state without considering obstacles.
4) **Check Constraints:** Check whether the trajectory after the perturbation satisfies all constraints from the MPD problem. If not, discard the current perturbation.
5) **Check Gap Tolerance:** Check whether the gap distance is less than a given tolerance. If yes, report the solution; otherwise, go to Step 2 until a given number iterations is satisfied.

Since computation of the final state is extensively used in both evaluation and finite-difference gradient calculation of the gap distance in the optimization, its computation cost directly affects the effectness of the gap reduction algorithms.

In pure numerical gap reduction algorithms, the final state is calculated by integrating the perturbed control from the initial state. Since analytical integration is only available for few ODEs, expensive numerical integration is normally used to calculate the new final states. The computation time for final states in each iteration grows linearly with the number of integration steps along a trajectory. Furthermore, with the above characterization, it is difficult to enforce the constraints on the states during the optimization process.

Therefore, instead of using the pure numerical gap reduction, we describe in the following sections how to employ symmetries of the robotic systems to efficiently evaluate the final state so that the computation time for the final state is reduced to a constant-time (with respect to integration accuracy) operation. It will also be shown that constraints on states and inputs are naturally

[5]The distance function is required to be continuous. Its value is non-negative and will be zero if and only if two states are equal.

maintained.

B. A class of systems with symmetries on principal fiber bundles

Symmetry is a fundamental geometric property of many robotic systems. Identification, properties, and theorems of symmetry for general systems are out of the scope of this paper. Refer to [36] for details. We will only give a brief description of symmetry of robotic systems, and then limit the scope of our paper to a class of systems with symmetries on principal fiber bundles [37], and finally use the car system in (9) as an example of a system in this class.

Consider a Lie group $\mathcal{G}$, with identity element e, acting on the state of the system through the (left) action

$$\Psi : \mathcal{G} \times X \to X; \tag{14}$$

we will often use the shorthand

$$\Psi_g(x) := \Psi(g, x). \tag{15}$$

We call $\mathcal{G}$ a symmetry group for the system in (3) 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 $\Phi_{\tilde{u}}(x_i, \cdot)$ of control $\tilde{u}$ from state x_i which is a solution to (3), the trajectory $\Psi_g \circ \Phi_{\tilde{u}}(x_i, \cdot)$ is also a solution to (3) for all $g \in \mathcal{G}$. It can be verified that invariance implies that group actions commute with state transitions. As shown in Fig. 6, if

$$x_{\mathrm{f}} = \Phi_{\tilde{u}}(x_{\mathrm{init}}, \bar{t}(\tilde{u})), \tag{16}$$

then

$$\Psi_g(x_{\mathrm{f}}) = \Phi_{\tilde{u}}(\Psi_g(x_{\mathrm{init}}), \bar{t}(\tilde{u})), \tag{17}$$

i.e.,

$$\Psi_g \circ \Phi_{\tilde{u}}(\cdot, \bar{t}(\tilde{u})) = \Phi_{\tilde{u}}(\cdot, \bar{t}(\tilde{u})) \circ \Psi_g. \tag{18}$$

Our paper considers a class of systems with symmetries on principal fiber bundles [37]. For clarity of exposition, we will assume that the state space X can be partitioned, at least locally, into the Cartesian product of two manifolds $X = \mathcal{G} \times \mathcal{Z}$, in which $\mathcal{Z} = \mathbb{R}^{n_z}$ with $n_z < n$ is the base space, and $\mathcal{G}$ is the fiber space; such an assumption is true in most cases of interest in robotics applications. Accordingly, we write the generic point $x \in X$ as the pair $(g, z) \in \mathcal{G} \times \mathcal{Z}$. We also require that the dynamics of these systems can be expressed in the following form:

$$\begin{aligned} \dot{g} &= g\xi(z) \\ \dot{z} &= f_z(z, u), \end{aligned} \tag{19}$$

in which $\xi(z)$ is a vector field corresponding to z and formulated as an element of $\mathfrak{g}$, the Lie algebra of $\mathcal{G}$. Such systems include many vehicles and moving robots, such as cars, submarines, and helicopters.

The car system with dynamics in (9) belongs to this class of systems and has symmetry group $SE(2)$. An element g of $SE(2)$ can be represented in homogeneous coordinates [38] as the following matrix

$$g = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix}, \tag{20}$$

which denotes either a transformation with θ-angle rotation and $[x, y]^T$-translation, or a configuration with position $[x, y]^T$ and orientation θ of a rigid body in

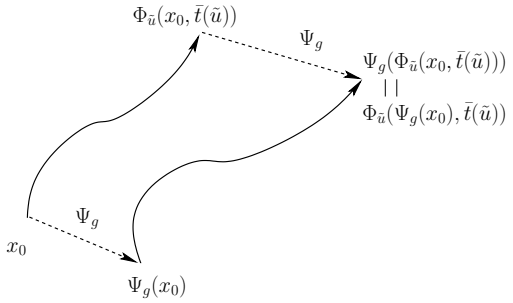

Fig. 6. The group actions commute with state transitions.

the plane. The state space X is partitioned into the Cartesian product of $SE(2) \times \mathbb{R}^2$, in which $SE(2)$ is the fiber space, and $\mathbb{R}^2$ is the base space. The state $x = (v_y, \omega, x, y, \theta)$ is written in the form (g, z), in which $g \in SE(2)$ represents (x, y, θ), and $z \in \mathbb{R}^2$ represents (v_y, ω). A group action $h \in SE(2)$ on state (g, z) is defined as

$$\Psi_h(g, z) = (hg, z). \tag{21}$$

It can be verified that the motion equation in (9) can be written in the form of (19) with

$$\xi(z) = \begin{bmatrix} 0 & -\omega & v_x \\ \omega & 0 & v_y \\ 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}(2), \tag{22}$$

in which $\mathfrak{se}(2)$ is the Lie algebra of $SE(2)$.

C. Coasting trajectories of the class of systems

For the system in (19), a state, $x = (g, z_0)$, is called a *coasting state* if there exists an input u_0, called a *coasting input*, such that

$$f_z(z_0, u_0) = 0. \tag{23}$$

A trajectory $\Phi_{\tilde{u}}(x_i, \cdot)$ from a coasting state $x = (g, z_0)$ is called a *coasting trajectory* if $\tilde{u}$ satisfies

$$\begin{aligned} \tilde{u}(t) &= u_0 \\ f_z(z_0, u_0) &= 0, \end{aligned} \tag{24}$$

for all $t \in [0, \bar{t}(\tilde{u})]$. One advantage of coasting trajectories is that the base variables and input are kept constant along the trajectory. If the constraints on the base variables and input are satisfied at the starting point, then they will also be satisfied along the trajectory if these constraints are independent of the fiber variables. Another advantage is that the fiber variables along the trajectory have the following explicit formulation:

$$g(t) = g(0) \exp(\xi(z_0)t), \tag{25}$$

in which "exp" denotes the group exponential [38] that maps the Lie algebra to the Lie group. When the Lie algebra and Lie group are both represented in the standard matrix form, the group exponential is the same as the matrix exponential. The coasting trajectory from the coasting state $x = (g, z_0)$ with coasting input u_0 and the constant control $\tilde{u}$ are

$$\begin{aligned} \Phi_{\tilde{u}}(x, t) &= (g \exp(\xi(z_0)t), z_0) \\ \tilde{u}(t) &= u_0, \end{aligned} \tag{26}$$

for all $t \in [0, \bar{t}(\tilde{u})]$. The final state of the coasting trajectory differs from its starting state by a group action

$$\Phi_{\tilde{u}}(x, \bar{t}(\tilde{u})) = \Psi_h(x), \tag{27}$$

in which

$$h = g \exp(\xi(z_0)t) g^{-1}. \tag{28}$$

For the car system in (9), the coasting state $x = (x, y, \theta, v_y, \omega)$ and coasting input u satisfy

$$\begin{aligned} 0 &= aC_{\mathrm{r}}(v_y - b\omega) + av_x^2 M\omega + bC_{\mathrm{r}}(v_y - b\omega) \\ u &= \tfrac{C_{\mathrm{r}}(v_y - b\omega) + v_x^2 M\omega + C_{\mathrm{f}}(v_y + a\omega)}{C_{\mathrm{f}} v_x} \\ u &\in U. \end{aligned} \tag{29}$$

Furthermore, since the constraints on u, v_y and ω are independent of fiber variables x, y and θ, the constraints are also satisfied along the coasting trajectory. There is an analytical solution for the final state of the coasting trajectory for the car since $\exp(\xi(z)t) \in SE(2)$ can be calculated analytically.

D. Efficient gap reduction with symmetry

For the class of systems considered in the paper, symmetry provides a decoupling between the base and fiber variables, i.e., the modification of the fiber variables of the starting state of a trajectory does not change the base variables of the trajectory. The decoupling provides us a way to eliminate the gap in the state space through two steps, which reduces the complexity of the gap

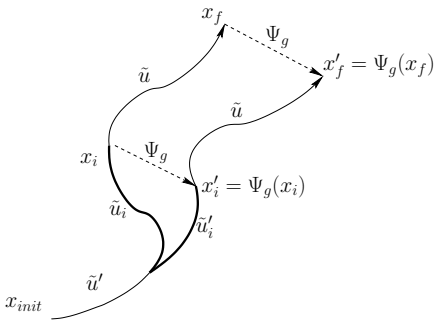

Fig. 7. Perturbation of $\tilde{u}_i$ with symmetry to avoid reintegration of $\tilde{u}$ in calculating $x'_{\rm f}$.

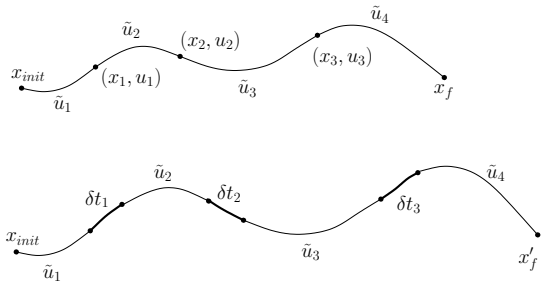

Fig. 8. Perturbation by inserting coasting trajectories.

reduction problem. In the first step, the gap in the base space is eliminated while ignoring the change of fiber variables. In the second step, the gap in the fiber space is eliminated. By the decoupling property, the base variables of the final state will be invariant in the second step, and therefore the gap in the state space is eliminated at the end of the second step. Since the gap reduction in the base space is system specific, we will provide examples for two systems in Section IV-A. In the following, the gap reduction in the fiber space will be provided as a general technique for different systems with symmetries.

1) Efficient final-state evaluation with symmetry: The idea of the efficient final state evaluation is shown in Fig. 7. If changing $\tilde{u}_i$ into $\tilde{u}'_i$ makes x_i differ from x'_i by a group action Ψ_g, then the new final state $x'_{\rm f}$ is obtained by applying Ψ_g on the old final state $x_{\rm f}$ without reintegrating $\tilde{u}$ from x'_i. In this paper, we achieve such a perturbation by inserting coasting trajectories described in Section III-C into the trajectory after coasting states.

Assume that the trajectory of $\tilde{u}$ in Fig. 8 has three coasting states x_1, x_2, x_3 associated coasting inputs u_1, u_2, and u_3, respectively. These three states divide $\tilde{u}$ into four controls $\tilde{u}_1$, $\tilde{u}_2$, $\tilde{u}_3$, and $\tilde{u}_4$, which satisfies

$$\tilde{u} = \tilde{u}_1 \circ \tilde{u}_2 \circ \tilde{u}_3 \circ \tilde{u}_4 \tag{30}$$

and have durations t_1, t_2, t_3, and t_4, respectively. Let

$$\Phi_{\tilde{u}}^{\bar{t}(\tilde{u})} := \Phi_{\tilde{u}}(\cdot, \bar{t}(\tilde{u})) : X \to X. \tag{31}$$

Since the motion equation is time-invariant, we have

$$x_{\rm f} = \Phi_{\tilde{u}_4}^{t_4} \circ \Phi_{\tilde{u}_3}^{t_3} \circ \Phi_{\tilde{u}_2}^{t_2} \circ \Phi_{\tilde{u}_1}^{t_1}(x_{\rm init}). \tag{32}$$

We can insert coasting trajectories of constant controls from these coasting states. These constant controls, denoted as $\tilde{u}'_1, \tilde{u}'_2$, and $\tilde{u}'_3$, have value u_1, u_2, and u_3 and durations δt_1, δt_2, and δt_3, respectively. The new final state is

$$\begin{aligned} x'_{\rm f} &= \Phi_{\tilde{u}_4}^{t_4} \circ \Phi_{\tilde{u}'_3}^{\delta t_3} \circ \Phi_{\tilde{u}_3}^{t_3} \circ \Phi_{\tilde{u}'_2}^{\delta t_2} \circ \Phi_{\tilde{u}_2}^{t_2} \circ \Phi_{\tilde{u}'_1}^{\delta t_1} \circ \Phi_{\tilde{u}_1}^{t_1}(x_{\rm init}) \\ &= \Phi_{\tilde{u}_4}^{t_4} \circ \Psi_{h_3} \circ \Phi_{\tilde{u}_3}^{t_3} \circ \Psi_{h_2} \circ \Phi_{\tilde{u}_2}^{t_2} \circ \Psi_{h_1} \circ \Phi_{\tilde{u}_1}^{t_1}(x_{\rm init}) \\ &= \Psi_{h_3} \circ \Psi_{h_2} \circ \Psi_{h_1} \circ \Phi_{\tilde{u}_4}^{t_4} \circ \Phi_{\tilde{u}_3}^{t_3} \circ \Phi_{\tilde{u}_2}^{t_2} \circ \Phi_{\tilde{u}_1}^{t_1}(x_{\rm init}) \\ &= \Psi_{h_3} \circ \Psi_{h_2} \circ \Psi_{h_1}(x_{\rm f}), \end{aligned} \tag{33}$$

in which

$$h_i = g_i \exp(\xi(z_i \delta t_i)) g_i^{-1}, \tag{34}$$

and

$$x_i = (g_i, z_i), i = 1, 2, 3. \tag{35}$$

In (33), the second line comes from (27), that is, the final state of the coasting trajectory differs from its starting state by a group action. The third line comes from (18). Note that the calculation of the new final state only

needs to evaluate the group actions of coasting trajectories without reintegrating the original controls. The group actions needs at most the numerical integration of the coasting trajectories, and could even be evaluated analytically when the symmetry group is the finite product of subgroups of $SE(3)$, e.g., $\mathbb{R}^1$, S^1, and SO^2. Furthermore, the new final state is fully parameterized by nonnegative durations of coasting trajectories.

Calculating the durations of the coasting trajectories to achieve the desired group action is called *trajectory planning via inverse kinematics* in [19], which generally requires numerical solutions except for few cases [39].

2) Selection of a subspace for optimization: Generally, if a gap exists in a space of dimension n_{g}, perturbating n_{g} parameters of the final state could eliminate the gap. If the number of parameters for the final state is larger than n_{g}, it is important to determine which n_{g} parameters could better eliminate the gap.

Observing that most gradient-based optimization techniques employ the steepest descent method as their starting step, it is expected that a nonlinear program with better convergence rate at the starting point for the steepest descent method might have a good chance to converge faster using other optimization techniques. Therefore, we calculate the convergence rate $\frac{1}{\alpha}$ of the steepest descent method [40], [41] for the nonlinear program at the starting point as follows:

$$\alpha^2 = 1 - \frac{(\sum_i \varsigma_i^2 \lambda_i^2)^2}{(\sum_i \varsigma_i^2 \lambda_i^3)(\sum_i \varsigma_i^2 \lambda_i)}, \qquad (36)$$

in which $\{\lambda_i\}$ are eigenvalues of Jacobian matrix J and $\{\varsigma_i\}$ are coefficients of linear decomposition of $x_{\mathrm{f}} - x_{\mathrm{goal}}$ with respect to eigenvectors of J. We choose multiple subsets of parameters and evaluate their convergence rates. The subset with the highest convergence rate is used for optimization.

The intuition of (36) is shown in Fig. 9. Assume that the final state has three parameters, and the gap is in a 2-dimensional space. Vectors v_1, v_2, and v_3 are the partial derivatives of the final state with respect to three parameters, respectively. Perturbation using parameters of v_2 and v_3 would be less likely to move the final state to the goal state than using parameters of v_1 and v_3, thus, the convergence rate of v_2 and v_3 would be less than that of v_1 and v_3, or that of v_1 and v_2.

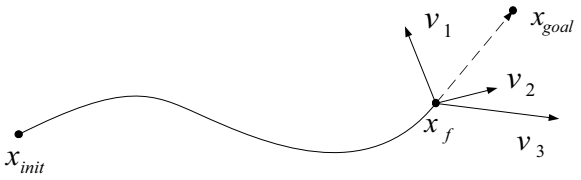

Fig. 9. The intuition of subspace selection with (36) for optimization.

E. Incorporating gap reduction with sampling-based MPD

To show the effect of gap reduction algorithms, we combine them with three types of sampling-based planning algorithms, which are respectively the unidirectional and bi-directional RRT-based planning algorithms [7], and the PRM-based planning algorithm [28].

The basic unidirectional RRT-based algorithm has been described in Section II-B. It can be extended into bi-directional search with minor modification. Initially, two RRTs, $\mathcal{T}_1$ and $\mathcal{T}_2$, are initialized with nodes n_{init} and n_{goal}, respectively. Nodes n_{init} and n_{goal} are associated with x_{init} and x_{goal}, respectively. Each tree is incrementally built similarly as the single tree, except that one of them is extended backward in time from the goal state. To check for possible solutions, every time a new node is generated from one tree, the distance between the new node and each node in the other tree is checked. If the distance between two nodes is less than ϵ_g, the two

RRTs are connected by unifying these two nodes, and the solution is the control of the path from the initial state node to the goal state node.

From the algorithm description for RRT-based planners, we can see that the gaps are only induced in Step 5 (described in Section II-B), to check whether the distance between two states is less than the gap tolerance. Given two states x_1 and x_2, and assuming that state x_1 is associated with node n_{f} in the tree containing the initial state node, the modification to check whether there is a solution path through these two states is as follows:

1) **Find Solution Path Candidate:** Check whether the distance between two states is less than a given large gap tolerance. If not, report no solution path passes through these two states; otherwise, we have a path candidate.
2) **Eliminate Base Space Gap:** Retrieve the control $\tilde{u}$ of the path from n_{init} to n_{f}. Check whether the gap in the base space between state x_2 and the final state x_1 of the trajectory of $\tilde{u}$ can be eliminated. If not, report that no solution path passes through these two states; otherwise, construct $\tilde{u}'$, so that the final state of its trajectory has the same base state as x_2.
3) **Eliminate Fiber Space Gap:** Use the perturbation method described in Section III-A to eliminate the gap between state x_2 and the final state of the trajectory of $\tilde{u}'$. If the gap distance is less than the given small gap tolerance, report that a solution path exists; otherwise, report that no solution path passes through these two states.

PRM-based planning algorithms have been successfully applied to solve many challenging motion planning problems without differential constraints. PRM-based planners have two phases. The construction phase builds a graph that captures the connectivity of the collision-free subset of the configuration space. In the query phase, the planners try to connect the initial and goal states to the search graph and find a path from the initial state to the goal state. An important part of the algorithm is to connect a state to other nearby states. Without differential constraints, the connection of two states is simple and efficient. However, for MPD problems, each connection corresponds to a challenging Two point Boundary Value Problem (TBVP). Since analytical solutions are only available for few cases, sampling-based single-query algorithms, such as RRT-based planners, could be used to generate connection trajectories for two given states. However, the gaps in these connecting trajectories could greatly degrade the quality of the returned solution, and the alternative of using small gap tolerance dramatically increases the running time. Therefore, we use the above sampling-based planning algorithms with gap reduction as the connection method for PRM-based planning method, which could be used for systems for which no efficient TBVP solution exists.

IV. Simulation Studies

In this section, results of a serial of simulations with different systems were collected to show the necessity and performance improvement of the proposed planning methods. Specifically, Section IV-A describes two systems used in our simulations. We specifically chose one nonholonomic system and a system with dynamics. Their respective base space gap reduction methods are also provided. In Section IV-B, it is shown that the running time of sampling-based MPD increases dramatically when the gap tolerance decreases, which strongly suggests the necessity of planning with gap reduction. Sections IV-C and IV-D show the benefits of using symmetry and subspace selection in our gap

reduction algorithm. Finally, in Sections IV-E and IV-F, the performance improvement of sampling-based MPD with our gap reduction algorithm is demonstrated with unidirectional, bi-directional, and PRM-based sampling-based MPD algorithms.

All simulations were done on a 2.0 Ghz PC running Linux. The NAG library is used to solve the optimization in the gap reduction. All the running time is reported in seconds.

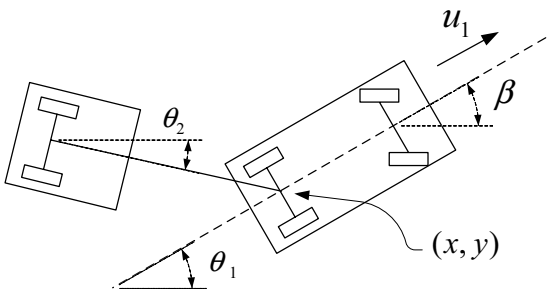

Fig. 10. Sketch of the car-and-trailer system.

A. Two systems used in simulations

The first system is the car with dynamics in (9). The base error of the car system is eliminated analytically as follows. The car system has the following linear base dynamics.

$$\dot{z} = Az + Bu, \tag{37}$$

in which

$$A = \begin{bmatrix} -\frac{C_\mathrm{f}+C_\mathrm{r}}{v_x M} & \frac{C_\mathrm{r}b-C_\mathrm{f}a}{v_x M} - v_x \\ \frac{C_\mathrm{r}b-C_\mathrm{f}a}{v_x I} & \frac{b^2C_\mathrm{r}+a^2C_\mathrm{f}}{v_x I} \end{bmatrix}, \tag{38}$$

and

$$B = \begin{bmatrix} -\frac{C_\mathrm{f}}{M} \\ \frac{C_\mathrm{f}a}{I} \end{bmatrix}. \tag{39}$$

Applying two constant controls with values c_1 and c_2 over two time intervals of the same duration, δt, from base variable z_1, we obtain that the final base variable z_2 as follows:

$$z_2 = A_\mathrm{f} z_1 + B_\mathrm{f} u_\mathrm{d}, \tag{40}$$

in which

$$A_\mathrm{f} = A_\mathrm{d}^2, \tag{41}$$

$$B_\mathrm{f} = \left[A_\mathrm{d}B_\mathrm{d}, B_\mathrm{d}\right], \tag{42}$$

$$A_\mathrm{d} = \exp(A\delta t), \tag{43}$$

$$B_\mathrm{d} = \int_0^{\delta t} \exp(A(\delta t - t))B dt, \tag{44}$$

and

$$u_\mathrm{d} = [c_1, c_2]^T. \tag{45}$$

Therefore, given base variables z_1 and z_2, we can calculate the value of two constant controls to eliminate the base space gap.

In some cases, the calculated input lies outside of the input space. To solve this, we iteratively double δt and repeat the above calculation until an admissible input is found.

The other one is the nonholonomic car-and-trailer system [21]. The system shown in Fig. 10 consists of a car pulling a trailer. Its state is $(x, y, \theta_1, \beta, \theta_2)$, in which $x \in [0, 400]$, $y \in [0, 400]$, and $\theta_1 \in [-\pi, \pi]$ are x-, y-coordinates and orientation of the car, $\beta \in [-0.6, 0.6]$ is the steering angle of the car, and $\theta_2 \in [-\pi, \pi]$ is the orientation of the trailer. The orientation θ_1 and θ_2 satisfy

$$|\theta_1 - \theta_2| < \frac{\pi}{2}. \tag{46}$$

The input to the system is (u_1, u_2), in which $u_1 \in [0, 2.0]$ is the forward velocity, and $u_2 \in [-0.24, 0.24]$ is the changing rate of the steering angle. Note, we intentionally set u_1 to be nonnegative so that the system is not STLC, and the gap cannot be reduced by trivially moving along the direction of vector fields generated by the Lie bracket. The motion equation of the car-and-

trailer system is:

$$\begin{aligned} \dot{x} &= u_1 \cos(\theta_1) \\ \dot{y} &= u_1 \sin(\theta_1) \\ \dot{\theta}_1 &= \frac{u_1 \tan(\beta)}{L_1} \\ \dot{\beta} &= u_2 \\ \dot{\theta}_2 &= \frac{u_1 \sin(\theta_1 - \theta_2)}{L_2}, \end{aligned} \tag{47}$$

in which $L_1 = 2.0$ is the length of the car, and $L_2 = 10.0$ is the length of the hitch. By introducing the transformation

$$\theta_{\rm d} = \theta_1 - \theta_2, \tag{48}$$

the last equation in (47) is changed to

$$\dot{\theta}_{\rm d} = u_1 \tan(\beta)/L_1 - u_1 \sin(\theta_{\rm d})/L_2. \tag{49}$$

The fiber variables are (x, y, θ_1), and the base variables are $(\beta, \theta_{\rm d})$. The coasting input and state satisfy

$$u_2 = 0, \tag{50}$$

and

$$\tan(\beta)/L_1 - \sin(\theta_{\rm d})/L_2 = 0. \tag{51}$$

Given base variables $(\beta, \theta_{\rm d})$ and $(\beta', \theta'_{\rm d})$ as the end points of the gap, we assume that

$$\theta'_{\rm d} > \theta_{\rm d} \tag{52}$$

without losing generality. The base-space gap of the car-and-trailer system can be eliminated in three steps: 1) Set u_2 to be the maximal value, and u_1 to be zero to increase β to its maximal value while keeping $\theta_{\rm d}$ constant. 2) Set u_2 be zero and u_1 be maximal to increase $\theta_{\rm d}$ to $\theta'_{\rm d}$ while keeping β constant. 3) Set u_2 be the minimal value and u_1 be zero to decease β to β' while keeping $\theta_{\rm d}$ constant.

B. Sensitivity of planner performance to the gap tolerance

To investigate the relationship between the performance of sampling-based planners without gap reduction and the gap tolerance, the bi-directional RRT-based planner [7] was used to solve a problem with the car with dynamics. The gap tolerances are selected to be 100.0, 10.0, 1.0, and 0.1, respectively. The problem is shown in Fig. 11, in which the initial and goal configurations are at the bottom-left and top-right corners, respectively, and the initial and goal velocities are zero. The distance between two gap endpoints in the simulations of the paper is calculated using the following function

$$\sum_{i=1}^{n} ||x_i, x'_i||^2 w_i \tag{53}$$

in which n is the dimension of the state space, $\{x_i\}$ and $\{x'_i\}$ are the state variables, and w_i is the weight for each dimension. The function $||x_i, x'_i||$ equals $\min(|x_i - x'_i|, 2\pi - |x_i - x'_i|)$ if x_i denotes the orientation, or $|x_i - x'_i|$ otherwise. For the 5-dimensional car, the weights are 1, 1, 1, 1, and 100, which are respectively for v_y, ω, x, y, and θ in (9). The weight for θ is specially chosen to be 100 since a small variation in orientation could greatly change the final state of a trajectory.

For each tolerance, twenty solution trials were executed. Each trial either returns a solution or reports failure after $400,000$ iterations. The running time and the number of returned solutions are shown in Table I, in which "Max.", "Min.", and "Avg." denote the maximal, minimal, and average time, and "Suc." denotes the number of successfully returned solutions over 20 runs. From the results, it can be seen that the performance of the sampling-based planners degrades dramatically with the gap tolerance decreasing. Note that with gap tolerances 1.0 and 0.1, the planner respectively only found 11 and 0 solutions over 20 runs as compared to 20 solutions out of 20 runs for gap tolerances 100.0 and 10.0. Therefore, there is no dramatic increase in running time when the gap tolerance respectively decreases from 10.0 to 1.0 and from 1.0 to 0.1. It is expected that the average time for

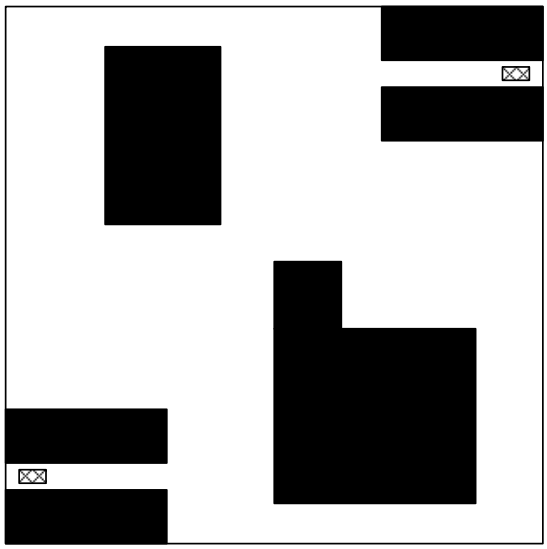

Fig. 11. A problem with the car with dynamics for bi-directional planners.

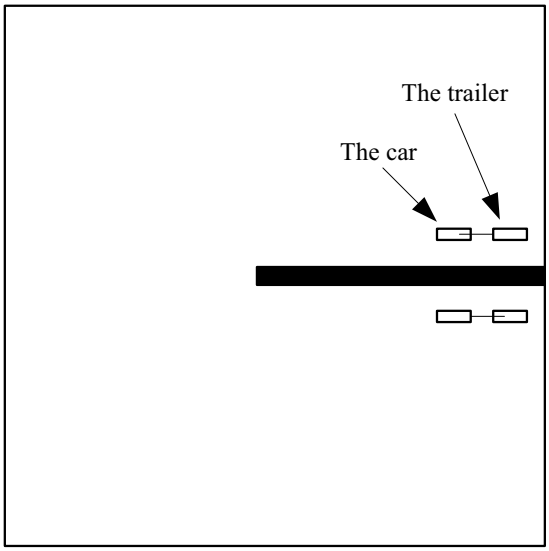

Fig. 12. A problem with the car-and-trailer for unidirectional planners, in which the initial and goal configurations are respectively above and below the black bar.

TABLE I

THE RUNNING TIME WITH DIFFERENT GAP TOLERANCE

ϵ_g	Max.	Min.	Avg.	Suc.
100.0	158.7	2.6	29.7105	20
10.0	7113.7	2.6	1992.7	20
1.0	60736.2	2075.8	30576.3	11
0.1	61785.7	60168.1	60736.2	0

gap tolerance 1.0 and 0.1 would be much bigger if the algorithm finds 20 solutions over 20 runs. Due to limited computational resources, we stopped the algorithms after $400,000$ iterations, instead of keeping running until a solution is found.

C. Comparisons of the classical numerical gap reduction and the gap reduction with symmetry

To compare the gap reduction algorithms with and without using symmetries, we also implemented the classical numerical gap reduction algorithm, in which the final states are evaluated by numerically integrating the perturbed controls. Both gap reduction algorithms were incorporated with unidirectional RRT-based planners and test on problems in Figures 3 and 12. For each problem, we ran twenty trials and reported the overall running time and number of numerical integrations. The gap tolerance is chosen to be 0.1 or $1.0e$–6. The weights of the gap distance for the car-and-trailer are 1, 1, 10, 1, and 10, which are respectively for x, y, θ_1, β, and θ_2 in (47).

Parameters for the final states of trajectories of controls from planners in [7] are as follows. Since the local planners in [7] use constant controls with fixed duration as sampled controls to generate new states. Therefore, each sampled control can be characterized by $m+1$ parameters, in which m of them denotes the value of the m-dimensional input and one is for the duration of the control. If a path candidate consists of k edges, then the control of the path will consist of $(m+1)k$ parameters.

The simulation results are shown in Table II, in which "Ob." denotes whether obstacles are considered, "Tra." denotes the problem with the car-and-trailer, "Sy." denotes whether symmetry is used, "Time" is the overall running time, "Num." is the overall number of integration, "Su." is the number of returned solutions over 20 runs. Observe that the planners using gap reduction with symmetry required much less time and number of numerical integration. Each numerical integration is over a fixed time interval of length 0.01 second. In the last row in the table, no overall running time and number of integration are obtained since the simulation is terminated when the planner using gap reduction without symmetry failed to return a solution in four days in one trial. The running time decreased respectively about 3 and 1000 times for problems involving the car with dynamics and the car-and-trailer system. In the analysis of the results, a large amount of trajectories returned from the classical gap reduction failed to be a solution because the local constraints on states along the trajectories are violated, especially for the constraints in (46) for the car-and-trailer system. It is the main reason that the planner using the gap reduction with symmetry has much better performance than that using the gap reduction without symmetry to solve the problems with the car-and-trailer system.

TABLE II

THE COMPARISON OF RUNNING TIME AND NUMBER OF INTEGRATION FOR PLANNERS WITH OR WITHOUT USING SYMMETRY

	Ob.	Sy.	ϵ_g	Time	Num.	Su.
Car	N	Y	0.1	753.0	8.2e7	20
Car	N	N	0.1	23091.3	4.5e9	20
Car	Y	Y	1.0e-6	36845.1	8.3e7	20
Car	Y	N	1.0e-6	94199.8	1.1e10	20
Tra.	N	Y	0.1	225.5	9.4e6	20
Tra.	N	N	0.1	243508.0	4.7e10	20
Tra.	Y	Y	1.0e-6	303.5	1.0e7	20
Tra.	Y	N	1.0e-6	-	-	-

TABLE III

THE EFFECTS OF SUBSPACE SELECTION

	Sel.	ϵ_g	Time	Num.	Suc.
Car	Y	1.0e-6	2395.3	1351	20
Car	N	1.0e-6	3678.0	5275	20
Tra.	Y	1.0e-6	457.0	3272	20
Tra.	N	1.0e-6	607.0	13304	20

D. Effects of subspace selection for optimization

To study the effect of subspace selection, one gap reduction algorithm was implemented for which the subspace is randomly generated. The planners using gap reduction with or without subspace selection were used to solve the problems with the car and the car-and-trailer in Figures 3 and 12 over 20 trials. To concentrate the comparison of the effects of subspace selection, obstacles in the problems are ignored. The overall running time, the number of calls for the NAG optimization function, and the number of returned solutions are reported in Table III, in which "Sel." denotes whether the subspace selection is used, "Time" is the overall running time, "Num." is the overall number of calls to the NAG optimization function, "Suc." is the number of returned solutions over 20 runs. From the table, observe that the overall time and number of calls in the planner using gap reduction with subspace selection are less than those in the planner using gap reduction without careful subspace selection.

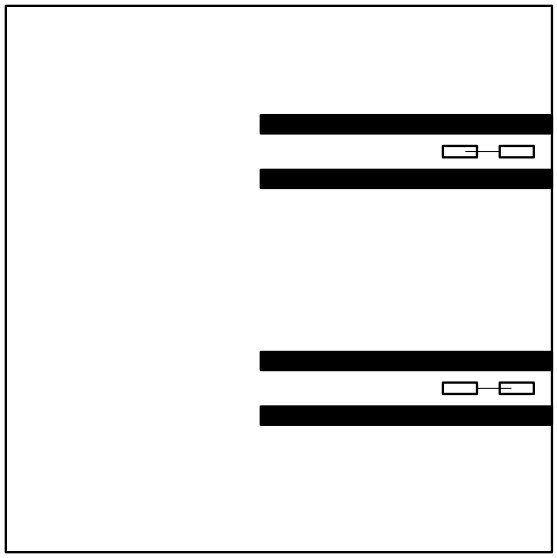

Fig. 13. A problem with the car-and-trailer for bi-directional planners, in which the initial and goal configurations are at the top and bottom half of the picture.

E. Performance improvements of unidirectional and bi-directional planners

The gap reduction algorithm was combined with the unidirectional and bi-directional RRT-based planner and tested on problems in Figures 3, 12, 11 and 13. The gap tolerance is set to be $1.0e$–6. For each problem, the original planner and the improved planner using gap reduction with symmetry tried twenty runs. The running time and number of returned solutions are shown in Tables II and IV, in which "Time" denotes the overall running time, and "Suc." denotes the number of returned solution over 20 runs. Note that the running time for original planners was not reported since no solution with gap tolerance $1.0e$–6 was reported for 20 runs.

TABLE IV

THE RESULTS OF USING THE IMPROVED BI-DIRECTIONAL PLANNERS TO SOLVE PROBLEMS

	ϵ_g	Time	Suc.
Car	1.0e-6	4294.2	20
Trailer	1.0e-6	15888.2	20

F. A PRM-based sampling-based planner with gap reduction

The gap reduction algorithm was combined with the basic PRM-based planner and test on the problem in Fig. 14 with the car model in (9). The gap tolerance was chosen to be $1.0e$–6. The construction and query processes alternated as follows. Every time, after the construction process inserted into the roadmap 50 new vertices, we will query solutions for 40 randomly chosen initial and goal state pairs. To reduce the construction time, a sampling point tries to connect to at most 20 neighbors and each connection runs for $2,000$ iterations in the construction phase. In the query phase, a sampling point tries to connect to at most 40 neighbors and each connection runs for $20,000$ iterations to fully utilize the constructed roadmap. The results are shown in Table V, in which "V.N." means the number of vertices, "C.T." is the construction time, "Q.T." and "Suc." are respectively the overall query time and number of returned solutions for 40 queries, and "E.N." is the number of edges in the roadmap. We can see that the PRM-based method could be used to solve MPD problems even an analytical steering method is not available.

V. CONCLUSION

In this paper, we identified the gap problem in sampling-based MPD algorithms, which is also experimentally verified that the problem causes dramatic

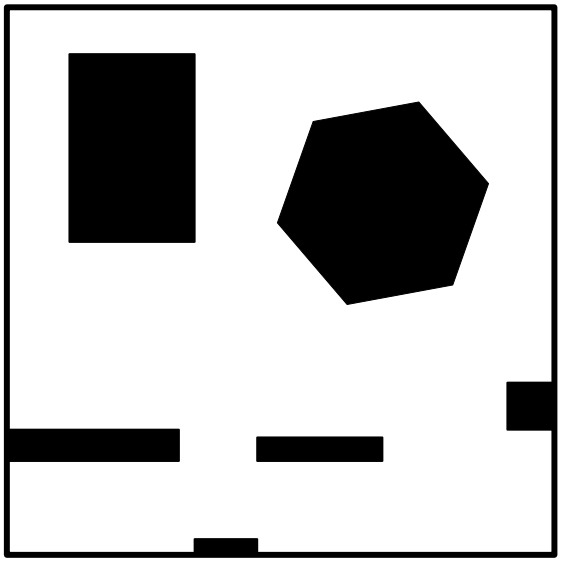

Fig. 14. A problem with the car model for PRM-based planners.

TABLE V

SIMULATION RESULTS FOR THE PRM-BASED PLANNER

V.N.	C. T.	Q. T.	Suc.	E.N.
50	2243.3	45333.5	5	32
100	7847.6	57520.4	7	154
150	17387.5	32817.7	13	363
200	26507.0	88226.3	18	586
250	40331.9	138218.0	31	795

increase in the algorithm running time when the gap tolerance decreases. To overcome this problem, an improvement for a broad class of planners using gap reduction with symmetry was proposed. By combining our gap reduction algorithm with unidirectional and bidirectional MPD algorithms, we calculated high quality solutions with small gap tolerances, which cannot be solved in reasonable time using original planners. Furthermore, using an improved planner as the connection method, we designed a PRM-based MPD algorithm and obtained promising results in solving a problem with a system which has no analytical solution for TBVPs.

One problem of the proposed algorithm is that obstacles are ignored in gap reduction. A resulting trajectory after successfully eliminating the gap could be in collision. Note that the smallest distance between the states along the trajectory and obstacles could be obtained through the collision checking algorithm. A possible extension of the algorithm could be using these distance information to consider collision avoidance in gap reduction. If the size of the variation of the states along the trajectory due to perturbation is less than the smallest distance to the obstacle, then the perturbed trajectory will be collision-free. Another interesting direction is about how to achieve local adjustments to the trajectory without changing the initial and goal states. A promising application of this extension is for reactive systems. Given a precalculated trajectory for the reactive system, local adjustments to the trajectory could be used for the system to avoid changing obstacle constraints.

APPENDIX

Parameters for the car with realistic dynamics are as follow: $M = 100$ (slug). Slug is an English unit of mass and commonly used in the vehicle industry. One slug is about 32.1 lbs. $C_{\mathrm{f}} = 17000.0$, $C_{\mathrm{r}} = 20000.0$, $a = 4.0$ (feet), $b = 5.0$ (feet), and $I = 1600.0$ (slug·foot^2).

ACKNOWLEDGMENTS

We thank Florent Lamiraux, Jean-Paul Laumond and Stephen Lindemann for helpful discussions. 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).

REFERENCES

[1] J.-P. Laumond, "Feasible trajectories for mobile robots with kinematic and environment constraints," in *ECAI*, 1986.

[2] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic planning," *Journal of the ACM*, vol. 40, pp. 1048–1066, Nov. 1993.

[3] J. T. Schwartz and M. Sharir, "On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Communications on Pure and Applied Mathematics*, vol. 36, pp. 345–398, 1983.

[4] S. Sastry, *Nonlinear systems, analysis, stability and control*. New York, NY: Springer, 1999.

[5] M. Akinc, K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plakue, and L. E. Kavaki, "Probabilistic roadmaps of trees for parallel computation of multiple query roadmaps," in *11th Int. Symp. Robot. Res.*, 2003.

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

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

[8] P. Choudhury and K. Lynch, "Trajectory planning for second-order underactuated mechanical systems in presence of obstacles," in *Workshop on Algorithmic Foundations of Robotics*, 2002.

[9] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

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

[11] S. LaValle and J. K. Jr., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[12] K. Lynch and M. Mason, "Stable pushing: Mechanics, controllability, and planning," *Int. J. Robot. Res.*, vol. 15, no. 6, pp. 533–556, 1996.

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

[14] J. Reif and H. Wang, "Non-uniform discretization approximations for kinodynamic motion planning," in *Algorithms for Robotic Motion and Manipulation*, J.-P. Laumond and M. Overmars, Eds. Wellesley, MA: A K Peters, 1997, pp. 97–112.

[15] 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.

[16] E. Frazzoli, "Robust hybrid control for autonomous vehicle motion planning," Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, June 2001.

[17] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with primitives," *IEEE Trans. on Robotics*, Oct. 2005, (to appear).

[18] D. Balkcom and M. Mason, "Time optimal trajectories for bounded velocity differential drive vehicles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 199–217, Mar. 2002.

[19] F. Bullo and K. Lynch, "Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 4, pp. 402–412, 2001.

[20] L.Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[21] R. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *Trans. Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

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

[23] P. Rouchon, M. Fliess, M. Levine, and P. Martin, "Flatness, motion planning, and trailer systems," in *Proc. IEEE Conf. on Decsion and Control*, 1993, pp. 2700–2705.

[24] 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.

[25] ——, "Improving the performance of sampling-based planners by using a symmetry-exploiting gap reduction algorithm," in *IEEE Int. Conf. Robot. & Autom.*, 2004.

[26] F. Lamiraux, E. Ferre, and E. Vallee, "Kinodynamic motion planning: Connecting exploration trees using trajectory optimization methods," in *IEEE Int. Conf. Robot. & Autom.*, 2004, pp. 3987–3992.

[27] F. Lamiraux, D. Bonnafous, and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," *IEEE Trans. Robot. & Autom.*, vol. 20, no. 6, pp. 967–977, 2004.

[28] L.Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. & Autom.*, vol. 12, no. 4, pp. 566–580, June 1996.

[29] P. Cheng, Z. Shen, and S. LaValle, "RRT-based trajectory design for autonomous automobiles and spacecraft," *Archives of Control*

Sciences, vol. 11, pp. 51–78, 2001.

[30] J. Bernard, J. Gruening, and K. Hoffmeister, "Evaluation of vehicle/driver performance using genetic algorithms," in *Society of Automotive Enfineers*, no. 980227, 1998.

[31] A. Ungoren and H. Peng, "Evaluation of vehicle dynamic control for rollover prevention," *International Journal of Automotive Technology*, vol. 5, no. 2, pp. 115–122, 2004.

[32] S. M. LaValle, *Planning Algorithms*. [Online], 2004, available at http://msl.cs.uiuc.edu/planning/.

[33] J. Pearl, *Heuristics*. Reading, MA: Addison-Wesley, 1984.

[34] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," in *The Fourth International Workshop on Algorithmic Foundations of Robotics*, 2000.

[35] A. M. Ladd and L. E. Kavraki, "Fast exploration for robots with dynamics," in *Proc. Workshop on Algorithmic Foundation of Robotics*, 2004.

[36] F. Bullo and A. Lewis, *Geometric Control of Mechanical Systems*. Springer Verlag, 2004.

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

[38] R. Murray, Z. Li, and S. Sastry, *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[39] 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*, 2003, pp. 625–630.

[40] D. G. Luenberger, *Linear and Nonlinear Programming*. New York: Springer, 2003.

[41] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Available from "http://www-2.cs.cmu.edu/∼jrs/jrspapers.html", 1994.