Exploiting Group Symmetries to Improve Precision in Kinodynamic and Nonholonomic Planning

Peng Cheng* Emilio Frazzoli† Steven M. LaValle*

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

Abstract

We address the problem of eliminating gaps in paths that are constructed by some nonholonomic and kinodynamic motion planning algorithms. In many of these algorithms, control inputs at each planning step are chosen from a finite set, obtained from discretization of the available control input set. While this approach is attractive for computational reasons, it can generate gaps, or discontinuities, either between path segments or between the final state and the desired goal. For the purpose of reducing gaps, the original control set and continuous time interval can be utilized, and incremental perturbations may be applied to incrementally optimize the gap error while respecting collision constraints. By exploiting Lie group symmetries, which emerge in a broad class of robot systems, we are able to avoid costly numerical integrations that usually occur in each step of gradient-based optimization techniques. It is hoped that the approach can ultimately lead to faster planning algorithms by allowing coarser discretizations of time and the available input set, with the understanding that later refinements can be made efficiently.

1 Introduction

A fundamental robotics problem is the automatic construction of control laws that lead to collision-free paths that satisfy kinematic and possibly dynamic equations of motion. Therefore, considerable efforts have been put forth to develop algorithms that compute trajectories for nonlinear systems with non-convex constraints. Techniques that apply to the largest class of problems usually require discretization and numerical approximation to a continuous-time and continuous-input system. For example in [3, 7–9, 17, 20, 25], numerical dynamic programming yields approximately-optimal solutions. Other recent approaches, such as [5, 10, 11, 21, 23], aim at the construction of trajectories through the sequential combination of several "motion primitives" from a finite library, chosen through a process of quantization of feasible control input histories. Other sampling-based techniques have also been proposed, particularly for solving high dimensional problems [13,16,18]. Besides our work, other recent interest in optimizing computed paths for nonholonomic systems includes [15].

Most of the existing methods that apply to general nonlinear systems either: 1) incrementally build a search graph from the initial state toward the goal state, or 2) incrementally build two trees, one from the initial state and another from the goal state, with hopes that the two trees will eventually meet. To avoid searching the same state repeatedly and to terminate the search in a finite number of iterations, many methods have been used to discretize the configuration or state space such that there are only a finite number of nodes in the search graph. In [9], the discretization is based on a grid built from the acceleration bounds and a fixed time step. Nonuniform boxes are used to discretize the state space in [25]. The configuration space is decomposed into disjoint parallelepipeds of equal size and asymptotic completeness is derived in [3].

Although methods based on discretization and numerical computation often apply to broad classes of problems, a common limitation is the occurrences of gaps in the solution. For example, an effect of the quantization of available control inputs is that the system is not small-time locally controllable (STLC): the reachable sets under such control laws has in several cases the structure of a lattice, thus preventing the exact matching of the final point of the trajectory with an arbitrary goal, in a finite number of steps. (In cases in which the reachable set is everywhere dense [22], or even continuous [10], 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.) Single-tree approaches usually will not achieve an exact goal state, and the quality of the solution strongly depends on the level of quantization that was used in the algorithm. Dual-tree, bidirectional approaches can precisely attain the initial and goal states; however, there is usually a gap in the middle of the solution where the two trees meet.

One is often faced with the tradeoff of selecting more-expensive approximations of the original system, either

*Dept. of Computer Science.
†Dept. of Aeronautical and Astronautical Engineering.

of its state space, or of the set of control inputs, for the purpose of reducing this error. If efficient methods can be developed to dramatically reduce this error, then the solution quality would be greatly improved. This is especially true for trajectories for which a gap appears in the middle, since trajectory integration from an erroneous starting point could dramatically alter the final state. Furthermore, the efficiency of numerical approaches to trajectory design and nonholonomic planning can be greatly improved by attempting to solve problems at a very coarse level of resolution with the understanding that substantial refinement to reduce gaps can be applied at any point. Efficient gap reduction techniques can also be used to enable multiple-query roadmap methods, such as those in the *probabilistic roadmap* framework [1, 2, 4, 12, 19, 24, 28, 30], to be adapted to problems that involve differential constraints. Although some such methods already exist [26, 29], they typically require a steering solution to be applied to avoid the gaps that would arise when connections are attempted between states. If a gap-reduction technique is efficient enough, it could enable the applicability of this framework to a much broader class of systems.

2 Problem Formulation

The focus in this paper is on a subproblem that typically arises in kinodynamic and nonholonomic planning. First, we formulate the general motion planning problem, and then we formally state the subproblem of interest in this paper, i.e., the improvement of the precision of the motion planning solution.

The planning problem The planning problem is expressed as a seven-tuple, $(\mathcal{X}, \rho, \mathcal{D}, x_{init}, x_{goal}, \mathcal{U}, f)$, in which: 1) $\mathcal{X}$ is a differentiable manifold of dimension n; 2) $\rho : \mathcal{X} \times \mathcal{X} \rightarrow [0, \infty)$ is a metric on $\mathcal{X}$; 3) $\mathcal{D} : \mathcal{X} \rightarrow \mathbb{R}$ represents collision constraints on $\mathcal{X}$ by requiring that the state, x, satisfy $\mathcal{D}(x) \leq 0$; 4) x_{init} represents an initial state; 5) x_{goal} represents a desired goal state; 6) $\mathcal{U} \subset \mathbb{R}^m$ represents a compact set of inputs or controls; 7) f is a smooth mapping from $\mathcal{X} \times \mathcal{U}$ to the tangent bundle $T\mathcal{X}$ that encodes the system dynamics as a set of time-invariant Ordinary Differential Equations (ODEs),

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

in which $x(t) \in \mathcal{X}$, $u(t) \in \mathcal{U}$, and $t \in [0, \infty)$. For nonholonomic planning, f represents non-integrable constraints on the tangent bundle $T\mathcal{X}$, and for kinodynamic planning, f includes second order time derivatives of the configuration variables (it may be an example of both nonholonomic and kinodynamic planning).

Let u denote a piece-wise continuous, open-loop control law, $u : [0, t_f) \rightarrow \mathcal{U}$, which can be used to produce a state, $x(t)$, via integration of (1), if some $x(0)$ is given and $t \in [0, t_f)$. Let $\Phi_u^t : \mathcal{X} \rightarrow \mathcal{X}$ denote a mapping that takes a starting state, $x(t_0)$, to a state $x(t_0 + t)$, after the application of u for duration t (note that the time argument to u will be shifted by $-t_0$ because the domain of u is $[0, t_f)$). The choice of t_0 is arbitrary due to time invariance of (1), and therefore it is not represented in the notation Φ_u^t.

The planning problem is to find some u (and associated t_f) such that $x_{goal} = \Phi_u^{t_f}(x_{init})$ and $\mathcal{D}(\Phi_u^t(x_{init})) \leq 0$ for all $t \in [0, t_f)$. In practice, however, quantization is performed in most planning algorithms, which leads to approximate goal satisfaction, $\rho(\Phi_u^{t_f}(x_{init}), x_{goal}) < \epsilon$, for some specified precision $\epsilon > 0$. Such a gap can appear, for instance, at the end of the trajectory using the grid-based dynamic programming method of [3], or even in the middle of the trajectory when trying to connect two trees in the bidirectional Rapidly-exploring Random Trees (RRT) approach [18]. In this paper, we only formulate the problem of reducing the gap at the end of a trajectory; however, only minor notational adjustments are necessary to apply the techniques to reducing gaps in the middle of a trajectory.

In most cases, the running time of the planning algorithm will increase dramatically as ϵ is decreased. Therefore, it is useful to solve the subproblem of taking a computed u, and perturbing it into a new control function that achieves improved accuracy by reducing the gap between $\Phi_u^{t_f}(x_{init})$ and x_{goal}. By this approach, a planning algorithm can generate a solution quickly for a large ϵ tolerance, and then improve the accuracy in a second stage, which is the main topic of this paper.

Closing Gaps to Improve Precision Assume here that a control function u has been computed as an approximate solution to the planning problem, which is specified by the seven-tuple. The task is to perturb u into a control function, $v : [0, t_f') \rightarrow \mathcal{U}$, such that

$$\rho(\Phi_v^{t_f'}(x_{init}), x_{goal}) \ll \rho(\Phi_u^{t_f}(x_{init}), x_{goal}),$$

while ensuring that $\mathcal{D}(\Phi_u^t(x_{init})) \leq 0$ for all $t \in [0, t_f')$. This optimization of ρ will occur as a sequence of small perturbations that incrementally transform u into v while preserving all constraints. Ideally, we would like to obtain $\Phi_v^{t_f'}(x_{init}) = x_{goal}$, but this will not usually be possible due to numerical considerations and limitations of a particular homotopy class (our methods will not be allowed to change path classes).

Perturbations to u will be generally achieved by first partitioning the domain of u into k small segments, and then reparameterizing each segment to start at time 0 while preserving its length. Choose k positive real values, t_1, t_2, ..., t_k, such that $t_1 + t_2 + \cdots t_k = t_f$. For each $i \in \{1, \ldots, k\}$, a function $u_i : [0, t_i) \rightarrow \mathcal{U}$ is constructed so that $u_i(t) = u(t_1 + \cdots + t_{i-1} + t)$. Due to time-invariance, note that the same state will be reached whether u is applied or u_1, ..., u_k are applied in succession. In terms of Φ, this can be expressed as

$$\Phi_u^{t_f}(x_{init}) = \Phi_{u_k}^{t_k} \circ \cdots \circ \Phi_{u_2}^{t_2} \circ \Phi_{u_1}^{t_1}(x_{init}). \tag{2}$$

Suppose that a small perturbation is made to some u_i. To determine the resulting final state, numerical inte-

gration must ordinarily be carried out using u_j for all j such that $i \geq j \geq k$.

Numerical optimization algorithms require the frequent evaluation of the distance function ρ with respect to various new control functions. Direct application of standard optimization techniques will lead to excessive computational requirements in all but the most trivial cases (e.g. linear or low-dimensional systems).

Our optimization methods perform perturbations that avoid the costly numerical integrations by exploiting fundamental geometric symmetry properties of the systems of interest in robotic motion planning. The group actions are used to quickly transform the latter portions of a changed trajectory, which avoids integrations and leads to dramatic performance improvements.

3 Symmetries of Robot Systems

The main idea behind the algorithm we propose is the exploitation of the symmetries in the system dynamics. In this section we define and illustrate symmetries, i.e., invariance properties of mechanical systems, and some of their applications to motion planning.

For the sake of simplicity, we assume that the state can be partitioned, at least locally, into the Cartesian product of two manifolds $\mathcal{X} = \mathcal{G} \times \mathcal{Z}$, where $\mathcal{G}$ is a Lie group with identity element e. For simplicity of exposition, we will assume that $\mathcal{Z} = \mathbb{R}^{n_z}$, $n_z < n$. Accordingly, we write the generic point $x \in \mathcal{X}$ 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 $\mathcal{X}$ is a principal fiber bundle; see [14].

Let the left action, denoted as Ψ, of $\mathcal{G}$ onto $\mathcal{X}$ be defined as follows:

$$\begin{aligned} \Psi : \mathcal{G} \times \mathcal{X} &\to \mathcal{X} \\ (h, x) &\mapsto \Psi_h(x) = (hg, z). \end{aligned}$$

We call $\mathcal{G}$ a symmetry group for the system (1) if its dynamics are *invariant* with respect to the left action. Invariance is equivalent to the following statement. Given any trajectory $t \mapsto (x(t), u(t)) \in \mathcal{X} \times \mathcal{U}$ which is a solution to equation (1), the trajectory $t \mapsto (\Psi_h(x(t)), u(t))$ is also a solution to equation (1) for all $h \in \mathcal{G}$.

Generally, the dynamics of systems with symmetry group $\mathcal{G}$ can be decomposed in the following way:

$$\begin{aligned} \dot{g} &= f_g(g, z) \\ \dot{z} &= f_z(z, u). \end{aligned} \tag{3}$$

It can be easily verified that invariance implies that group actions commute with state transitions; from the above definition of invariance, we know that if $x_f = \Phi_u^{t_f}(x_{init})$, then $\Psi_h(x_f) = \Phi_u^{t_f}(\Psi_h(x_{init}))$, i.e., $\Phi_u^{t_f} = \Psi_h^{-1} \circ \Phi_u^{t_f} \circ \Psi_h$. This leads to the key step in our proposed procedure. When we modify the input signal u_i of duration t_i to u_i' of duration t_i', we will require that for some h_i in $\mathcal{G}$, the corresponding state transition, at the i-th step, can be decomposed as follows:

$$\Phi_{u_i'}^{t_i} = \Psi_{h_i} \circ \Phi_{u_i}^{t_i}. \tag{4}$$

Thus, the perturbed state transition can be decomposed as an action of the symmetry group on the unperturbed state transition. If this is true, then the perturbed final state x_f' , given a change in the control signal during the i-th interval only, can be expressed as

$$\begin{aligned} x_f' &= \Phi_{u_k}^{t_k} \circ \cdots \circ \Phi_{u_i'}^{t_i'} \circ \cdots \circ \Phi_{u_1}^{t_1}(x_{init}) \\ &= \Phi_{u_k}^{t_k} \circ \cdots \circ \Psi_{h_i} \circ \Phi_{u_i}^{t_i} \circ \cdots \circ \Phi_{u_1}^{t_1}(x_{init}) \\ &= \Psi_{h_i} \circ \Phi_{u_k}^{t_k} \circ \cdots \circ \Phi_{u_i}^{t_i} \circ \cdots \circ \Phi_{u_1}^{t_1}(x_{init}) \\ &= \Psi_{h_1}(x_f). \end{aligned} \tag{5}$$

Thus, we need to compute only the new state transition during interval i to determine the new final position.

Intuitively, the fact that the perturbed state transition at the i-th step differs from the original state transition by an action of the symmetry group, means that the remaining part of the trajectory can be "translated" rigidly without need for re-computation.

Enforcing (4) is equivalent to requiring that the perturbation does not change the starting and ending condition, in each time interval, of the base variable z. In other words, if $x(t) = (g(t), z(t))$ is the nominal trajectory, and $x_p(t) = (g_p(t), z_p(t))$ is the perturbed trajectory, we require than $z(T_i) = z_p(T_i)$ with $T_i = \sum_{j=1}^{i} t_i$, for $i = 1 \ldots k$.

4 Using Symmetries to Avoid Integrations

Because computation of the final state is extensively used in both evaluation and finite-difference gradient calculation of the objective function, we will concentrate on how to calculate the final state efficiently by exploiting symmetries of the system. Without this method, the computation time in each iteration grows linearly with the number of integration steps along a trajectory. By exploiting symmetries, the computation time is reduced to a constant-time operation (with respect to integration accuracy). In this section, two methods are presented: one based on feedback-linearizable base dynamics, and another for systems with affine-in-control base dynamics. Our simulations in Section 6 indicate that the second of these methods is by far the most efficient.

Method 1: Feedback-linearizable base dynamics A system with feedback-linearizable base dynamics is one for which there exists a change of coordinates $(z, u) \mapsto (\zeta, v)$ for system equation (3) such that the dynamics on the transformed base are linear, i.e., equation (3) can be rewritten as:

$$\begin{aligned} \dot{g} &= f_g(g, \zeta) \\ \dot{\zeta} &= A\zeta + Bv. \end{aligned} \tag{6}$$

Systems which can be reduced to this form include many static- and dynamic-feedback-linearizable systems . The unicycle model (10) and 5-dimensional car

model (11), on which we did simulations, belong to this category. See Section 6.1.

To modify the trajectory and maintain the base at time T_i, we employ the linear structure on the base of these systems (6). Specifically, for time interval t_i with control signal u_i, we could partition it into l subintervals of equal length $\delta t_i = \frac{t_i}{l}$. Assume the control signal u_i^j in subinterval j is constant for $j = 1, \ldots, l$ and the base of the starting state of interval t_i is z_{i-1}. Then we can calculate the base of the end state z_i of interval t_i with a closed form by solving the linear time-invariant ODEs describing the base dynamics.

$$z_i = A_f z_{i-1} + B_f u_d, \tag{7}$$

where $A_f = A_d^l$, $B_f = \left[A_d^{l-1}B_d, A_d^{l-2}B_d, \cdots, B_d\right]$, $A_d = e^{A\delta t_i}$, $B_d = \int_0^{\delta t_i} e^{A(\delta t_i - t)} B dt$, and $u_d = [(u_i^1)^T, \cdots, (u_i^l)^T]^T$. Therefore, if we choose v from the kernel space of B_f in (7), the base state will be maintained at time T_i and the final state will be calculated efficiently.

Method 2: Affine-in-control base dynamics Even though the above method saves numerical integration in unperturbed time intervals, the computation is still expensive for the remaining numerical integration. The basic idea of the method introduced in this paragraph is that, instead of partitioning time intervals and choosing v in the kernel space of B_f in (7), we insert new trajectory intervals between two pre-existing intervals. However, to avoid numerical integration and exploit symmetries of the system, there should exist a control signal $\bar{u}$ for the inserted trajectory such that base will be kept constant, i.e., the $f_z(z, \bar{u}) = 0$ in (3).

Determining values of the control input u that satisfy this condition is particularly easy for systems with affine-in-control base dynamics, i.e., systems whose base dynamics is described by an ODE of the form

$$\dot{z} = f_x(x) + f_u(x)u.$$

Clearly, systems with feedback-linearizable base dynamics belong to this class.

An example of the system with affine-in-control base dynamics but without feedback-linearizable base dynamics is a trailer system used in our simulation. The trailer system consists of a car pulling a trailer. The state of the system can be represented using a local chart, as $(x, y, \theta_1, \beta, \theta_2)$, in which x, y, and θ_1 are x-, y-coordinates and orientation of the car, β is the steering angle of the car, 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 translational velocity along x-axis of the local frame of the car, and u_2 is the rate of change of the steering angle. The dynamics of the trailer system are as follows:

$$\begin{aligned} \dot{x} &= u_1 \cos(\theta_1) \\ \dot{y} &= u_1 \sin(\theta_1) \\ \dot{\theta}_1 &= u_1 \tan(\beta)/L_1 \\ \dot{\beta} &= u_2 \\ \dot{\theta}_2 &= u_1 \sin(\theta_1 - \theta_2)/L_2, \end{aligned} \tag{8}$$

in which L_1 is the length of the car, L_2 the length of the hitch. By introducing the transformation $\theta_d = \theta_1 - \theta_2$, the last equation in (8) is changed to $\dot{\theta}_d = u_1 \tan(\beta)/L_1 - u_1 \sin(\theta_d)/L_2$. Then, the fiber coordinates are (x, y, θ_1), and the base coordinates are (β, θ_d). Choosing $u_2 = 0$ at state with $\tan(\beta)/L_1 - \sin(\theta_d)/L_2 = 0$ will keep the base constant.

Assume that base at time T_i is $z(T_i) = \bar{z}$, a new trajectory interval $t \in (T_i, T_i + \delta t) \mapsto (x(t), \bar{z})$ is inserted after T_i. When the new trajectory has some control signal $\bar{u}$ to keep the base of the system constant, $\Phi_{\bar{u}}^{\delta t}$ is an element of the symmetry group, generated as the exponential of an element of the Lie algebra $\mathfrak{g}$ of $\mathcal{G}$. The element η of the Lie algebra is calculated from base $\bar{z}$.

5 General Algorithm Overview

Assume that a vector p in $\mathbb{R}^{k_p}$ characterizes the control function u, and is the input to the gap reduction algorithm. The k_p generally is large even when the control function is discretized over time. We define the final state as a function $x_f : \mathbb{R}^{k_p} \to \mathcal{X}$ according to (2). Reducing the gap is to search in $\mathbb{R}^{k_p}$ for a p^* such that $\rho(x_f(p^*), x_{goal}) \leq \rho(x_f(p), x_{goal})$ for all $p \in \mathbb{R}^{k_p}$ and the control function u, parameterized by p^*, is violation-free. By saying a control signal $u : [0, t_f) \to U$ is violation-free, we mean that $\mathcal{D}(\Phi_u^t(x_{init})) \leq 0$ for all $t \in [0, t_f)$. This gap reduction problem generally is a high-dimensional nonlinear program with nonvex constraints, in which $x_f(p)$ is a nonconvex function, nonconvex constraints exists due to obstacles in the configuration space and high-dimensionality comes from the large number k_p. In this paper, this problem is attacked by solving a sequence of low-dimensional nonlinear programs with nonconvex constraints. In each low-dimensional nonlinear program, a low-dimensional subspace of $\mathbb{R}^{k_p}$ is searched to optimize the objective function.

Outline of the algorithm The gap reduction algorithm is shown in Fig. 1. The algorithm runs iteratively until *NotTerminated*() returns "false", which means that the algorithm has run for either a given number of iterations or a given period of time. In each iteration, a low-dimensional subspace $\mathcal{S} \subset \mathbb{R}^{k_p}$ is first chosen. The subspace $\mathcal{S}$ combines with the objective function to compose a low-dimensional nonlinear problem, which is solved to generate a new parameter vector p_n. If the control function, parameterized by p_n, is not violation-free, p_n is discarded; otherwise, the value of $\rho(x_f(p_n), x_{goal})$ is checked. If the value is less than the given ϵ_c, p_n will be reported as the final solution; otherwise, current parameter vector p is set to p_n.

Selection of subspace for the nonlinear program Because of the combinatorial complexity of possible choices of subspace in $\mathbb{R}^{k_p}$, it is impractical to enumerate all choices. An easy way to select search subspace is to just randomly choose k_c dimensions from $\mathbb{R}^{k_p}$. To

avoid the calculation of high-dimensional gradient vectors, it is better to choose smaller k_c. However, because the perturbation of one parameter will cause the final state to move in a space of at most one dimension, k_c should be at least the dimension of the fiber space to ensure the fiber of the final state to move in the whole fiber space. The problem with the randomly chosen subspace $\mathcal{S}$ is that the Jacobian matrix J of the objective function with respect to variable elements of p in $\mathcal{S}$ might be ill-conditioned and/or the current p might be a bad starting point for the optimization. In these cases, the convergence rate of the optimization process could be very small, especially when the search space is restricted in the kernel space of B_f in (7).

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 approximately calculate the convergence rate $\frac{1}{\alpha}$ of the steepest descent method [27] for the nonlinear program at the starting point as follows:

$$\alpha^2 = 1 - \frac{(\sum_i \xi_i^2 \lambda_i^2)^2}{(\sum_i \xi_i^2 \lambda_i^3)(\sum \xi_i^2 \lambda_i)}, \tag{9}$$

in which $\{\lambda_i\}$ are eigenvalues of Jacobian matrix J and $\{\xi_i\}$ are coefficients of linear decomposition of $x_f(p) - x_{goal}$ with respect to eigenvectors of J.

Therefore, we choose a finite collection $\mathcal{C}$ of subspace of $\mathbb{R}^{k_p}$ and calculate α for each corresponding nonlinear program. The subspace with the smallest α is used to compose a nonlinear program. If the cardinality of $\mathcal{C}$ is large, a large amount of time in subspace selection will decrease the overall performance. If the cardinality of $\mathcal{C}$ is small, optimizing the chosen nonlinear programs with small convergence rate will also affect the performance. According to our experiences, when the cardinality of $\mathcal{C}$ was in $[\frac{k_p}{10}, \frac{k_p}{4}]$, the performance was the best.

6 Simulation Studies

We performed simulations based on three nonlinear systems: a unicycle, a car with dynamics, and a kinematic car pulling a trailer. The implementation was done in Matlab on a 1.2Ghz PC running Windows XP. Matlab's nonlinear programming function, fmincon, was used to solve low-dimensional nonlinear programs. Only the relative times are important in Fig. 3; we expect that the running times would be one or two orders of magnitude faster if implemented in C++ on a current PC under Linux.

```
Gap_Reduction(X, ρ, D, x_init, x_goal, U, f, p, ε_c)
 1  Initialize the algorithm;
 2  while NotTerminated() = true
 3      S := SubspaceSelection(R^{k_p});
 4      p_n := Optimize(ρ(x_f(p), x_goal)), p ∈ S;
 5      if Collision(p_n) = false then
 7          if ρ(x_f(p_n), x_goal) < ε_c then
 8              return p_n;
 9          else p := p_n;
10  return failure.
```

Figure 1: The gap-reduction algorithm.

6.1 Models

The unicycle A coordinate representation of the state is $(x, y, \theta, v_x, v_y, \omega)$, in which $x \in [0, 100]$, $y \in [0, 100]$, and $\theta \in [-\pi, \pi]$ represent position and orientation of the unicycle; $\omega \in [-3, 3]$ is the angular velocity, and $v_x \in [-15, 15]$ is translational velocity along the x-axis of the body frame. The translational velocity along the y-axis of the body frame is constrained to be zero. The body frame used to describe models in the paper is a coordinate fixed on the system moving in a 2D plane with x-axis along the forward direction and y-axis along the direction perpendicular to the forward direction in the plane. The input vector is (u_1, u_2), in which $u_1 \in [-1, 1]$ is the rate of change of v_x, and $u_2 \in [-4, 4]$ is the rate of change of ω. The equations of motion (f from Section 2) are

$$\begin{aligned} \dot{x} &= v_x \cos(\theta) \\ \dot{y} &= v_x \sin(\theta) \\ \dot{\theta} &= \omega \\ \dot{v}_x &= u_2 \\ \dot{\omega} &= u_1. \end{aligned} \tag{10}$$

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

$$\begin{aligned} \dot{v}_y &= -v_x\omega + (f_{yf}(u) + f_{yr}(u))/M \\ \dot{\omega} &= (f_{yf}a - f_{yr}b)/I \\ \dot{x} &= v_x \cos(\theta) - v_y \sin(\theta) \\ \dot{y} &= v_x \sin(\theta) + v_y \cos(\theta) \\ \dot{\theta} &= \omega, \end{aligned} \tag{11}$$

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 linear functions of u and represent forces acting on front and rear tires along the y-axis of the local frame.

The trailer system The equations of motion are given in (8). The state space bounds are: $x \in [0, 400]$, $y \in [0, 400]$, $\theta_1 \in [-\pi, \pi]$, $\beta \in [-0.6, 0.6]$, and $\theta_2 \in [-\pi, \pi]$. The bounds on inputs are $u_1 \in [0, 2.0]$ and $u_2 \in [-0.24, 0.24]$. Note, we intentionally set u_1, 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.

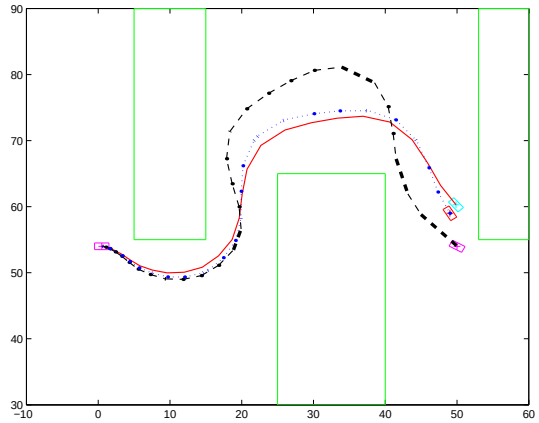
6.2 Results

Three methods were compared, which included a classical gradient descent, Method 1, and Method 2. The classical method does not use symmetries of the system and calculates the final state by numerically integrating the control function. All three methods used the randomized subspace selection algorithm from Section 5 with k_c be 6 and the cardinality of $\mathcal{C}$ be 10. Ten trials were done to the same problem for each algorithm; each trial consists of 20 iterations of subspace selection, and corresponding nonlinear program optimization. The average final value of objective functions and overall running time over ten trials were reported. When applying Method 2 on the trailer system, instead of inserting intervals after states with $\tan(\beta)/L_1 - \sin(\theta_d)/L_2 = 0$ to maintain the base, we inserted intervals after states with $\tan(\beta)/L_1 - \sin(\theta_d)/L_2 < 0.4$. After a new control function is returned from the gap reduction algorithm, we calculated final states of the new control function using symmetries and numerical integration, respectively. The only difference between two final states was at the orientation of the trailer, θ_2, and the maximum difference was 0.082.

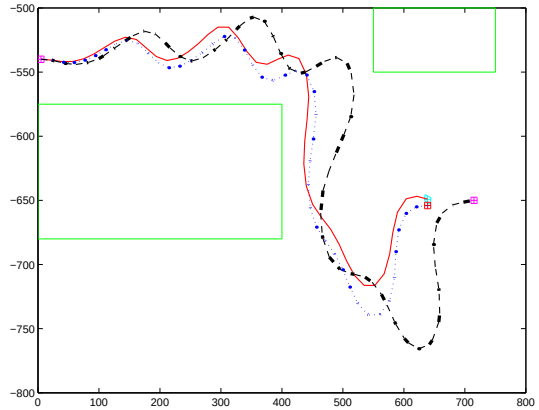
The objective function used to measure gap error between the final state and the goal state is $\rho(x_f, x_{goal}) = \sum_{i=1}^{n} w_i |x_f, x_{goal}|_i^2$, in which w_i is a weight for each dimension, and $|\cdot,\cdot|_i$ calculates the distance between the i^{th} element of x_f and x_{goal} considering topology. Weights are used for giving greater importance to state variables that may have smaller ranges. In our implementation, if the i^{th} element of the state corresponds to the orientation of the system, $w_i = 10$ and $|x, y|_i = \min(|x_i - y_i|, |x_i - y_i| - \pi)$; otherwise, $w_i = 1$ and $|x, y|_i = |x_i - y_i|$, in which $x, y \in \mathbb{R}^n$, x_i and y_i are i^{th} element of x and y, and $|\cdot|$ returns the absolute value.

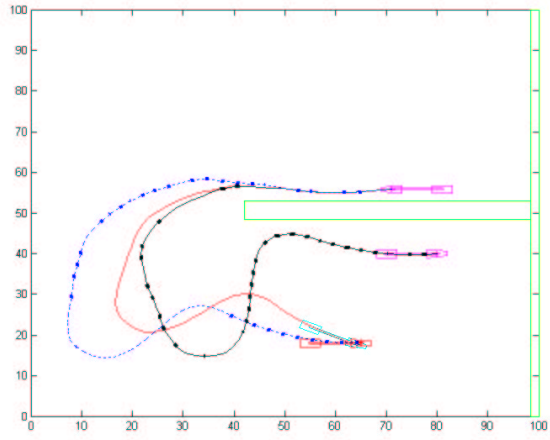
A piecewise constant control function for each motion planning problem, shown in Fig. 2, was calculated using the resolution-complete RRT algorithm [6]. The original gaps between final states and goal states are small in the sense that the initial value of objective functions is less than 5. We also changed the goal state to generate problems with large gaps, i.e., problems with the initial value of objective functions larger than 5. In fact, to make the problem more difficult, the initial value of objective functions for large gaps in Fig. 2 is much larger than 5.

All three methods were applied on each problem with both small and large gap. Sample solutions for each problem obtained using Method 2 are shown in Fig. 2. From simulations, we observed that all three methods could close small gaps and their performance is compared in Fig. 3. For problems with large gaps, because only Method 2 works, we provided its results separately to avoid excessive empty entries in Fig. 3.



1. Problem with the unicycle.





3. Problem with the trailer system.

Figure 2: Gap reduction for problems with three different models, in which solutions returned from a motion planner are shown as continuous lines; dashed lines show solutions after closing the large gaps; dotted lines show solutions after closing the small gaps, and thick segments along dashed and dotted lines are inserted trajectory intervals by the gap reduction algorithm.

	Ini. val.	Classical (w/o symm.)		Method 1		Method 2	
		av. f.v.	T (s)	av. f.v.	T (s)	av. f.v.	T (s)
1	2.6	0.040	3146	0.48	1654	0.013	39
2	3.9	0.062	9689	0.45	9191	0.043	97
3	4.9	0.21	65610	N/A	N/A	0.051	142

Figure 3: Comparison of performance for a classical approach, and two methods that exploit symmetries. Method 2 yields the best performance by far.

Comparison of performance on problems with small gaps Results of simulations is shown in Fig. 3, in which the numbers 1, 2 and 3, denote problems that involve the unicycle, car with dynamics, and the trailer system, respectively; "ini. val." is the initial value of objective functions; "av. f.v." is the average final value of objective functions; "T" is the overall running time. The data of Method 1 for the trailer system is not available (N/A) because the system is not feedback-linearizable. The duration of initial control functions from the planner for each problem is 13, 9.2, and 130 seconds, respectively

¿From the table, note that the performance of Method 1 and the classical method is comparable because even though Method 1 has smaller running time, its average final value is bigger than the latter method. There are two reasons. The first reason is that the initial control functions for problem 1 and 2 only last for 9.2 and 13 seconds, respectively. Because Method 1 perturbs the control function in the kernel space of B_f in (7), it sustains more constraints and a harder optimization problem which might need to solve more low-dimensional nonlinear programs. If the control function is short and the cost of numerical integration from x_{init} is not big, computational saving from the numerical integration might not cover the cost to solve more nonlinear programs. The second reason is that the input space of these two problems is small, $[-0.6, 0.6]$ and $[-1, 1] \times [-4, 4]$, respectively, which leads to an even harder optimization. We redesigned the problem with the unicycle model. The input space was increased to $[-10, 10] \times [-40, 40]$, and the length of the original control sequence was increased to 39 seconds. Ten trials were done using each method. Using the classical method, the overall running time is 17128 seconds. Using Method 1, the overall running time is 2633 seconds, which is significantly improved. Using Method 2, the overall running time is 187 seconds, which is dramatically superior.

Problems with large gaps Because only Method 2 worked for problems with large gaps, only results for Method 2 are presented.

For the problem with the unicycle, x_{init} is $(0.5, 54, 0, 0, 0)$, x_{goal} is $(50, 54, -0.5, 7, 0.4)$, and the initial final state x_f is $(49.9, 60, -0.8, 7, 0.4)$. Of ten trials, the average final value of the objective function is 0.021, and the overall running time is 73.4 seconds. For the problem with the car with dynamics, x_{init} is $(0, 0, 5, -540, 0)$, x_{goal} is $(7, 0.2, 715, -650, 0)$, and the initial x_f is $(7, 0.2, 639, -649, -0.25)$. Of ten trials, the average final value is 0.023, and the overall running time is 120.47 seconds. For the problem with the trailer system, x_{init} is $(71, 56, \pi, 0, \pi)$, x_{goal} is $(80, 40, 0, 0.04, 0)$, and the initial x_f is $(64, 18, -0.35, 0.04, -0.4)$. Of ten trials, the average final value is 0.0094, and the overall running time is 163.75 seconds.

7 Conclusions

Based on our experiments, we conclude that our symmetry-based methods are highly successful at reducing precision errors in computed trajectories. The methods apply to base dynamics that are feedback linearizable or affine in control, which includes many interesting systems that arise in robotics. With an optimized implementation, we expect the running times to be fast enough to enable many interesting improvements of existing nonholonomic and kinodynamic planning techniques. For example, coarser quantizations may be used, paths from multiple search trees can be combined, and entire search trees may be recycled by applying group actions. In future work, we hope to combine the methods developed here with planning algorithms in applications such as robotics and computer-generated animation.

Acknowledgments This work was funded in part by the following grants: NSF CAREER 0133869 (Frazzoli), NSF CAREER 9875304 (LaValle), NSF 0208891 (Frazzoli and LaValle), and NSF 0118146 (Bullo and LaValle).

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, pages 155–168, 1998.

[2] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 113–120, 1996.

[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] V. Boor, N. H. Overmars, and A. F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *IEEE Int. Conf. Robot. & Autom.*, pages 1018–1023, 1999.

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

[6] P. Cheng and S. M. LaValle. Resolution complete rapidly-exploring random trees. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 267–272, 2002.

[7] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chain manipulators. *Algorithmica*, 14(6):480–530, 1995.

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

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

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

[11] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. 2003.

[12] C. Holleman and L. E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *IEEE Int. Conf. Robot. & Autom.*, pages 1408–1413, 2000.

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

[14] 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.

[15] F. Lamiraux, D. Bonnafous, and C. Van Geem. Path optimization for nonholonomic systems: Application to reactive obstacle avoidance and path planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 3099–3104, 2002.

[16] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[17] 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.

[18] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 473–479, 1999.

[19] P. Leven and S. Hutchinson. Real-time motion planning in changing environments. In *Proc. International Symposium on Robotics Research*, 2000.

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

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

[22] 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.

[23] 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.

[24] C. Pisula, K. Hoff, M. Lin, and D. Manoch. Randomized path planning for a rigid body based on hardware accelerated Voronoi sampling. In *Proc. Workshop on Algorithmic Foundation of Robotics*, 2000.

[25] 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.

[26] 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.

[27] 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.

[28] T. Simeon, J.-P. Laumond., and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics Journal*, 14(6), 2000.

[29] 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.

[30] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *IEEE Int. Conf. Robot. & Autom.*, pages 1024–1031, 1999.