Provably Feasible Semi-Infinite Program Under Collision Constraints via Subdivision

Duo Zhang[1], Chen Liang[2], Xifeng Gao[1], Kui Wu[1], and Zherong Pan[1†]

***Abstract*—We present a semi-infinite program (SIP) solver for trajectory optimizations of general articulated robots. These problems are more challenging than standard Nonlinear Program (NLP) by involving an infinite number of non-convex, collision constraints. Prior SIP solvers based on constraint sampling cannot guarantee the satisfaction of all constraints. Instead, our method uses a conservative bound on articulated body motions to ensure the solution feasibility throughout the optimization procedure. We further use subdivision to adaptively reduce the error in conservative motion estimation. Combined, we prove that our SIP solver guarantees feasibility while approaching the optimal solution of SIP problems up to arbitrary user-provided precision. We demonstrate our method towards several trajectory optimization problems in simulation, including industrial robot arms and UAVs. The results demonstrate that our approach generates collision-free locally optimal trajectories within a couple of minutes.**

***Index Terms*—Semi-Infinite Program, Trajectory Optimization, Collision Handling, Articulated Body**

I. Introduction

This paper deals with trajectory generation for articulated robots, which is a fundamental problem in robotic motion planning. Among other requirements, providing strict collision-free guarantees is crucial to a reliable algorithm, i.e., the robot body should be bounded away from static and dynamic obstacles by a safe distance at any time instance. In addition to feasibility, modern planning algorithms such as [1] further seek (local) optimality, i.e., finding trajectories that correspond to the minimizers of user-specified cost functions. Typical cost functions would account for smoothness [2], energy efficacy [3], and time-optimization [4]. Despite decades of research, achieving simultaneous feasibility and optimality remains a challenging problem.

Several categories of techniques have attempted to generate feasible and optimal trajectories. The most widely recognized sampling-based motion planners [5] and their optimal variants [6] progressively construct a tree in the robot configuration space and then use low-level collision checker to ensure collision-free along each edge of the tree. The optimal trajectory restricted to the tree can asymptotically approach the global optima. However, most of the low-level collision checkers are based on discrete-time sampling [7] and cannot ensure collision-free during continuous motion. Optimal sampling-based methods are a kind of zeroth-order optimization algorithm that does not require gradient information to guide trajectory search. On the downside, the complexity of finding globally optimal trajectories is exponential in the dimension of configuration spaces [8].

† indicates corresponding author. [1]LightSpeed Studios, Tencent. {zduozhang, xifgao, kwwu, zrpan}@global.tencent.com. [2]The Department of Computer Science, Yale University. dylan.liang@yale.edu

In parallel, first- and second-order trajectory optimization algorithms [9] have been proposed to utilize derivative information to bring the trajectory towards a local optima with a polynomial complexity in the configuration space dimension. Based on the well-developed off-the-shelf NLP solvers such as [10], trajectory optimization has been adopted to solve complex high-dimensional planning problems. Despite the efficacy of high-order techniques, however, dealing with collision constraints becomes a major challenge as the number of constraints is infinite, leading to SIP problems [11]. Existing trajectory optimizers for articulated robots are based on the exchange method [12], i.e., sampling the constraints at discrete time instances. Similar to the discrete-time sampling used in the collision checkers, the exchange method can miss infeasible constraints and sacrifice the feasibility guarantee. In their latest works, Marcucci *et al.* [13] propose an alternative approach that first identifies feasible convex subsets of the configuration space, and then searches for globally optimal trajectories restricted to these subsets. While their method can provide feasibility and optimality guarantees, expensive precomputations are required to identify the feasible subsets [14].

We propose a novel SIP solver with guaranteed feasibility. Our method is based on the discretization-based SIP solver [12]. We divide a robot trajectory into intervals and use conservative motion bound to ensure the collision-free property during each interval. These intervals further introduce barrier penalty functions, which guide the optimizer to stay inside the feasible domain and approach the optimal solution of SIP problems up to arbitrary user-specified precision. The key components of our method involve: 1) a motion bound that conservatively estimates the range of motion of a point on the robot over a finite time interval; 2) a safe line-search algorithm that prevents the intersection between the motion bound and obstacles; 3) a motion subdivision scheme that recursively reduces the error of conservative motion estimation. By carefully designing the motion bound and line-search algorithm, we prove that our solver converges within finitely many iterations to a collision-free, nearly locally optimal trajectory, under the mild assumption of Lipschitz motion continuity. We have also evaluated our method on a row of examples, including industrial robot arms reaching targets through complex environments and multi-UAV trajectory generation with simultaneous rotation and translation. Our method can generate safe trajectories within a couple of minutes on a single desktop machine.

II. Related Work

We review related works in optimality and feasibility of motion planning, SIP and its application in robotics, and various safety certifications.

A. Optimality and Feasibility of Motion Planning

We highlight three milestones in the development of motion planning frameworks: the trajectory optimization approach [15], the rapid-exploring random tree (RRT) [5], and its optimal variant (RRT-Star) [6]. Although the performance of these frameworks largely depends on the concrete choices of algorithmic components, a major qualitative difference lies in their optimality and complexity. Trajectory optimization ensures the returned trajectory is optimal in a local basin of attraction; RRT returns an arbitrary feasible trajectory without any optimality guarantee; while RRT-Star provides an asymptotic global optimality guarantee. With the stronger guarantee in terms of optimality comes significantly higher complexity in the dimension of configuration spaces. Based on off-the-shelf NLP solvers such as [16], the complexity of trajectory optimization is polynomial. The complexity of RRT relies on the visibility property [17], which is not directly related to the dimension. Unfortunately, the complexity of optimal sampling-based motion planning is exponential [8], which is unsurprising considering the NP-hardness of general non-convex optimization. As a result, nearly global optimality can only be expected in low-dimensional problems, while local optimality is preferred in practical, high-dimensional planning problems. Although there have been considerable efforts in reducing the cost of RRT-Star, including the use of bidirectional exploration [18], branch-and-bound [19], informed-RRT-Star [20], and lazy collision checkers [21], its worst-case complexity cannot be shaken.

In addition to optimality, providing a strict feasibility guarantee poses a major challenge for any of the aforementioned frameworks. For trajectory optimization, collision-constraints are formulated as differentiable hard constraints in NLP. However, since there are the infinite number of constraints, practical formulations [9, 22, 23] need to sample constraints at discrete time instances, which violate the feasibility guarantee. Even worse, many off-the-shelf NLP solvers [10] can accept infeasible solutions and then use gradient information to guide the solutions back to the feasible domain, which is not guaranteed to succeed, especially when either the robot or the environment contains geometrically thin objects. An exception is the feasible SQP algorithm [24] that ensures iteration-wise feasibility, but this algorithm is not well-studied in the robotic community. On the other hand, RRT, RRT-Star, and their variants require a low-level collision checker to prune non-collision-free trajectory segments. The widely used discrete-time collision checker [7] again requires temporal sampling and violates the feasibility guarantee. There exists exact continuous-time collision checkers [25, 26, 27], but they make strong assumptions that robot links are undergoing linear or affine motions, which are only valid for point, rigid, or car-like robots. For more general articulated robot motions, inexact continuous-time collision checkers [28, 29, 30] have been proposed that provide motion upper bounds, but such bounds can be overly conservative and result in false negatives. In comparison, our trajectory optimization method also relies heavily on motion upper bounds, but we use recursive subdivision to adaptively tighten the motion bounds and provide both local optimality and feasibility guarantee for general configuration spaces.

B. SIP and Applications in Robotics

SIP models mathematical programs involving a finite number of decision variables but an infinite number of constraints. SIPs frequently arise in robotic applications for modeling constraints on motion safety [11, 14], controllability and stability [31, 32], reachability [33], and pervasive contact realizability [34]. The key challenge of solving SIP lies in the reduction of the infinite constraint set to a computable finite set. To the best of our knowledge, a generally equivalent infinite-to-finite reduction is unavailable, except for some special cases [35, 36]. Therefore, general-purpose SIP solvers [12] rely on approximate infinite-to-finite reductions that transform SIP to a conventional NLP, which is then solved iteratively as a sub-problem. Two representative methods of this kind are the exchange and discretization methods. These methods sample the constraint index set to approximately reduce SIP to NLP. In terms of our collision constraints, this treatment resembles the discrete-time collision checker used in sampling-based motion planners. Unfortunately, even starting from a feasible initial point, general-purpose SIP solvers cannot guarantee the feasibility of solutions, which is an inherited shortcoming of the underlying NLP solver. Instead, we propose a feasible discretization method for solving the special SIP under collision constraints with a feasibility guarantee. Our method is inspired by the exact penalty formulation [37, 38], which reduces the SIP to a conventional NLP by integrating over the constraint indices. The exact penalty method can be considered as a third method for infinite-to-finite reduction, but the integral in such penalty function is generally intractable to compute. Our key idea is to approximate such integrals by subdivision without compromising the theoretical guarantees.

C. Planning Under Safety Certificates

Our discussion to this point focuses on general algorithms applicable to arbitrary configuration spaces, where a feasibility guarantee is difficult to establish. But exceptions exist for several special cases or under additional assumptions. Assuming a continuous-time dynamic system, for example, the Control Barrier Function (CBF) [39, 40] designs a controller to steer a robot while satisfying given constraints, but CBF is only concerned about the feasibility and cannot guarantee the steered robot trajectory is optimal. By approximating the robot as a point or a ball, the robot trajectory becomes a high-order spline, and tight motion bound can be derived to ensure safety. This approach is widely adopted for (multi-)UAV trajectory generation [35, 41], but it cannot be extended to more complex robot kinematics. Most recently, a novel formulation has been proposed by Amice *et al.* [14] to certify the feasibility of a positive-measure subset of the configuration space of arbitrary articulated robots. Their method relies on reformulation that transforms the collision constraint to a conditional polynomial positivity problem, which can be further combined with mixed-integer convex programming, as done in [13], to ensure feasibility. Compared to all these techniques, our feasibility

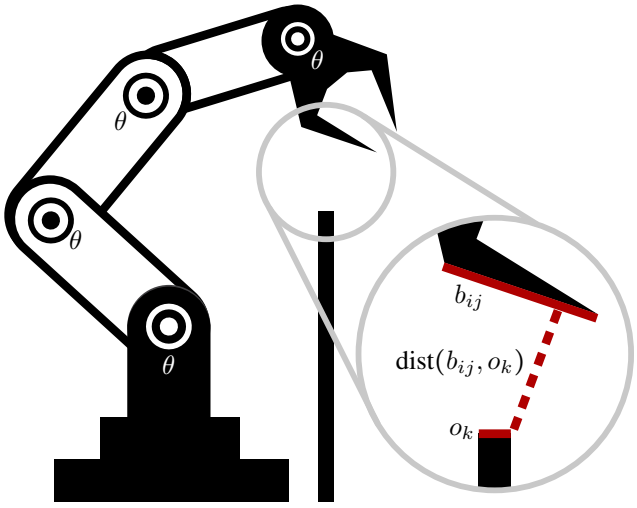


Fig. 1: We consider a moving articulated robot arm, where the volume occupied by the ith link is denoted as b_i. Each b_i admits a finite decomposition $b_i = \bigcup_j b_{ij}$ and each b_{ij} is a simple shape, e.g. the red edge. b_{ij} is a function of the time t and trajectory parameters θ, denoted as $b_{ij}(t, \theta)$. θ could be the control points of Bézier curves in the configuration space. Similarly, we can decompose the obstacle $o = \bigcup_k o_k$ where o_k is the short red edge. We introduce log barrier energy bounding the distance $\text{dist}(b_{ij}, o_k)$ (dashed line) away from a safety distance d_0.

guarantee is based on a much weaker assumption of Lipschitz motion bound, and we do not require any precomputation to establish the safety certificate.

III. Problem Statement

In this section, we provide a general formulation for collision-constrained trajectory generation problems. Throughout the paper, we will use subscripts to index geometric entities or functions, but we choose not to indicate the total number of indices to keep the paper succinct, e.g., we denote $\sum_i$ as a summation over all indices i. We consider an open-loop articulated robot as composed of several rigid bodies. The ith rigid body occupies a finite volume in the global frame, denoted as $b_i \subset \mathbb{R}^3$. Without ambiguity, we refer to the rigid body and its volume interchangeably. We further denote $b_i^0 \subset \mathbb{R}^3$ as the volume of ith body in its local frame. We further assume there is a set of static obstacles taking up another volume $o \subset \mathbb{R}^3$. By the articulated body kinematics, we can compute b_i from b_i^0 via the rigid transform: $b_i = M_i b_i^0$ where we define $M_i b_i^0 = \{M_i x | x \in b_i^0\}$. When a robot moves, M_i and thus b_i are time-dependent functions, denoted as $M_i(t, \theta)$ and $b_i(t, \theta)$, respectively. Here $t \in [0, T]$ is the time parameter and the trajectory is parameterized by a set of decision variables, denoted as θ. The problem of collision-constrained trajectory generation aims at minimizing a twice-differentiable cost function $\mathcal{O}(\theta)$, such that each rigid body b_i is bounded away from o by a user-specified safe distance denoted as d_0 at any $t \in [0, T]$. Formally, this is defined as:

$$\begin{aligned} &\underset{\theta}{\text{argmin}}\ \mathcal{O}(\theta) \\ &\text{s.t.} \quad \text{dist}(b_i(t, \theta), o) \geq d_0 \quad \forall i \wedge t \in [0, T], \end{aligned} \tag{1}$$

where $\text{dist}(\bullet)$ is the shortest Euclidean distance between two sets. Under the very mild assumption of being twice-differentiable, the cost function $\mathcal{O}(\theta)$ can encode various user requirements for a "good" trajectory, i.e., the closedness between an end-effector and a target position, or the smoothness of motion. This is a SIP due to the infinitely many constraints, one corresponding to each time instance. Further, the SIP is non-smooth as the distance function between two general sets is non-differentiable. Equation (1) is a general definition incorporating various geometric representations of the robot and obstacles as illustrated in Figure 1.

A. Smooth Approximation

Although the main idea of this work is a discretization method for solving Equation (1), most existing SIP solvers already adopt the idea of discretization for spatial representation of a rigid body b_i to deal with non-smoothness of the function $\text{dist}(\bullet)$. By spatial discretization, we assume that b_i endows a finite decomposition $b_i = \bigcup_j b_{ij}$ where b_{ij} is the jth subset of b_i in world frame. Similarly, we can finitely decompose o as $o = \bigcup_k o_k$ where o_k is the kth subset of environmental obstacles o in the world frame. If the distance function $\text{dist}(b_{ij}, o_k)$ between a pair of subsets is differentiable, then we can reduce the non-smooth SIP Equation (1) to the following smooth SIP:

$$\begin{aligned} &\underset{\theta}{\text{argmin}}\ \mathcal{O}(\theta) \\ &\text{s.t.} \quad \text{dist}(b_{ij}(t, \theta), o_k) \geq d_0 \quad \forall i, j, k \wedge t \in [0, T]. \end{aligned} \tag{2}$$

In summary, spatial discretization is based on the following assumption:

Assumption III.1. *Each b_i and o endows a finite decomposition denoted as $b_i = \bigcup_j b_{ij}$ and $o = \bigcup_k o_k$ such that $\text{dist}(b_{ij}, o_k)$ is sufficiently smooth for any $\langle i, j, k \rangle$.*

Assumption III.1 holds for almost all computational representations of robot links. For example, common spatial discretization methods include point cloud, convex hull, and triangle mesh. In the case of the point cloud, each b_{ij} or o_k is a point, and $\text{dist}(\bullet)$ is the differentiable pointwise distance. In the case of the convex hull, $\text{dist}(\bullet)$ is the distance between a pair of convex hulls, which is non-differentiable in its exact form, but can be made sufficiently smooth by slightly bulging each convex hull to make them strictly convex [42]. In the case of the triangle mesh, it has been shown that the distance between a pair of triangles can be reduced to two sub-cases: 1) the distance between a point and a triangle and 2) the distance between a pair of edges, see [25], both of which are special cases of the convex hull. Although spatial discretization can generate many more distance constraints, only a few constraints in close proximity to each other need to be activated and forwarded to the SIP solver for consideration, and these potentially active constraints can be efficiently identified using

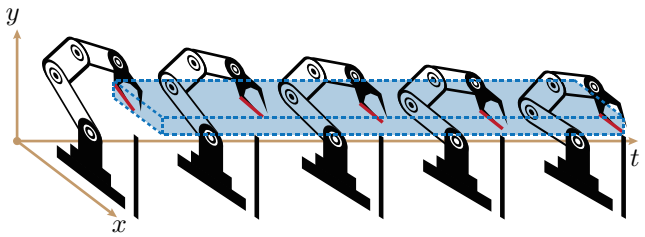


Fig. 2: When the red edge illustrated in Figure 1 is tracing out a temporal trajectory, we use a spatial-temporal motion bound (blue) to estimate its range and guarantee safety.

a spatial acceleration data structure [43]. Despite these spatial discretizations, however, the total number of constraints is still infinite in the temporal domain.

B. The Exchange Method

The exchange method is a classical algorithm for solving general SIP, which reduces SIP to a series of NLP by sampling constraints both spatially and temporally. Specifically, the algorithm maintains an instance set $\mathcal{I}$ that contains a finite set of $\langle i, j, k, t\rangle$ tuples and reduces Equation (1) to the following NLP:

$$\begin{aligned} &\underset{\theta}{\text{argmin}}\ \mathcal{O}(\theta) \\ &\text{s.t.}\quad \text{dist}(b_{ij}(t,\theta), o_k) \geq d_0 \quad \forall\, \langle i,j,k,t\rangle \in \mathcal{I}. \end{aligned} \tag{3}$$

The algorithm approaches the solution of Equation (1) by alternating between solving Equation (3) and updating $\mathcal{I}$. The success of the exchange method relies on a constraint selection oracle for updating $\mathcal{I}$. Although several heuristic oracles are empirically effective, we are unaware of any exchange method that can guarantee the satisfaction of semi-infinite constraints. Indeed, most exchange methods insert new $\langle i, j, k, t\rangle$ pairs into $\mathcal{I}$ when the constraint is already violated, i.e., $\text{dist}(b_{ij}(t,\theta), o_k) < d_0$, and relies on the underlying NLP solver to pull the solution back onto the constrained manifold, where the feasibility guarantee is lost.

IV. Subdivision-Based SIP Solver

We propose a novel subdivision-based SIP solver inspired by the discretization method [12]. Unlike the exchange method that selects the instance set $\mathcal{I}$ using an oracle algorithm, the discretization method uniformly subdivides the index set into finite intervals and chooses a surrogate index from each interval to form the instance set $\mathcal{I}$, reducing the original problem into an NLP. As a key point of departure from the conventional infeasible discretization method, however, we design the surrogate constraint in Section IV-A such that its feasible domain is a strict subset of the true feasible domain of Equation (1). We then show in Section IV-B and Section IV-C that, by using the feasible interior point method such as [44, Chapter 4.1] to solve the NLP, our algorithm is guaranteed to generate iterations satisfying all the surrogate constraints. Since our surrogate constraint can limit the solution to an overly conservative subset, in Section IV-D, we introduce a subdivision method to adaptively adjust the conservative subset and approach the original feasible domain.

A. Surrogate Constraint

We consider the following infinite spatial-temporal subset of constraints:

$$\text{dist}(b_{ij}(t,\theta), o_k) \geq d_0 \quad \forall t \in [T_0, T_1] \subseteq [0, T], \tag{4}$$

where b_{ij} and o_k are two spatial subsets and $[T_0, T_1] \subseteq [0, T]$ is a temporal subset. Our surrogate constraint replaces the entire time interval with a single time instance. A natural choice is to use the following midpoint constraint:

$$\text{dist}\left(b_{ij}\left(\frac{T_0+T_1}{2}, \theta\right), o_k\right) \geq d_0, \tag{5}$$

which is differentiable by Assumption III.1. Unfortunately, the domain specified by Equation (5) is larger than that of Equation (4), violating our feasibility requirement. We remedy this problem by upper-bounding the feasibility error due to the use of our surrogate. A linear upper bound can be established by taking the following mild assumption:

Assumption IV.1. *The feasible domain of t and θ is bounded.*

Lemma IV.2. *Under Assumption III.1, IV.1, there exists a constant L_1 such that:*

$$|dist(b_{ij}(t_1,\theta), o_k) - dist(b_{ij}(t_2,\theta), o_k)| \leq L_1 |t_1 - t_2|.$$

Proof. A differentiable function in a bounded domain is also Lipschitz continuous so that we can define L_1 as the Lipschitz constant. □

The above result implies that the feasibility error of the midpoint surrogate constraint is upper bounded by $L_1(T_1 - T_0)/2$. Further, the feasible domain is specified by the following more strict constraint:

$$\text{dist}(b_{ij}(t,\theta), o_k) \geq d_0 + L_1 |T_1 - T_0|/2,$$

is a subset of the true feasible domain. However, such a subset can be too restrictive and oftentimes lead to an empty feasible domain. Instead, our method only uses Lemma IV.2 as an additional safety check as illustrated in Figure 2, while the underlying optimizer deals with the standard constraint Equation (5). Note that the bound in Lemma IV.2 is not tight and there are many sophisticated upper bounds that converge superlinearly, of which a well-studied method is the Taylor model [29]. Although we recommend using the Taylor model in the implementation of our method, our theoretical results merely require a linear upper bound.

B. Barrier Penalty Function

To ensure our algorithm generates feasible iterations, we have to solve the NLP using a feasible interior-point method such as [44, Chapter 4.1]. These algorithms turn each inequality collision constraint into the following penalty function:

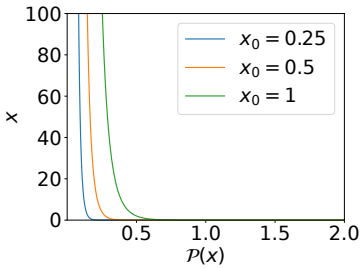
$$\mathcal{P}_{ijk}(t,\theta) \triangleq \mathcal{P}\left(\text{dist}\left(b_{ij}\left(t,\theta\right), o_k\right) - d_0\right),$$

where $\mathcal{P}$ is a sufficiently smooth, monotonically decreasing function defined on $(0, \infty)$ such that $\lim_{x\to 0} \mathcal{P}(x) = \infty$ and $\lim_{x\to\infty} \mathcal{P}(x) = 0$. In order to handle SIP problems, we need the following additional assumption to hold for $\mathcal{P}$:

Assumption IV.3. *The barrier function $\mathcal{P}$ satisfies:*

$$\lim_{x\to 0} x\mathcal{P}(x) = \infty.$$

The most conventional penalty function is the log-barrier function $\mathcal{P}(x) = -\log(x)$, but this function violates Assumption IV.3. By direct verification, one could see that a valid penalty function is $\mathcal{P}(x) = -\log(x)/x$. In [43], authors showed that a locally supported $\mathcal{P}$ is desirable for a spatial acceleration data structure to efficiently prune inactive constraints, for which we propose the following function:



$$\mathcal{P}(x) = \begin{cases} \frac{(x_0-x)^3}{x^4} & x \le x_0 \\ 0 & x > x_0, \end{cases}$$

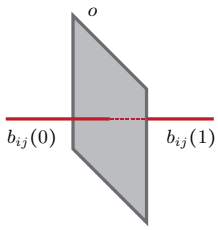
which is twice differentiable and locally supported within $(0, x_0]$ with x_0 being a small positive constant. The intuition behind Assumption IV.3 lies in the integral reformulation of semi-infinite constraints. Indeed, we can transform the infinite constraints into a finite form by integrating the penalty function over semi-infinite variables, giving the following finite integral penalty function, denoted as $\bar{\mathcal{P}}$:

$$\bar{\mathcal{P}}_{ijk}(T_0, T_1, \theta) \triangleq \int_{T_0}^{T_1} \mathcal{P}_{ijk}(t, \theta) dt. \tag{6}$$

The above integral penalty function has been considered in [37, 38, 45] to solve SIP. However, their proposed algorithms are only applicable for special forms of constraints, where the integral in Equation (6) has a closed-form expression. Unfortunately, such an integral in our problem does not have a closed-form solution. Instead, we propose to approximate the integral via spatial-temporal discretization. We will show that the error in our discrete approximation is controllable, which is crucial to the convergence of our proposed solver. In order for the penalty function to guarantee feasibility, $\bar{\mathcal{P}}_{ijk}$ must tend to infinity when:

$$\exists t \in [T_0, T_1] \quad \text{dist}\left(b_{ij}\left(t, \theta\right), o_k\right) \to d_0. \tag{7}$$

However, the log-barrier function does not satisfy this property. As illustrated in the inset, suppose there is a straight line trajectory $b_{ij}(t) = \left(t, 0, 0\right)$ along the positive X-axis, o is the YZ-plane that intersects the X-axis at $\left(1/2, 0, 0\right)$, $T = 1$ and $d_0 = 0$, then $\bar{\mathcal{P}}$ takes the following finite value:



$$\bar{\mathcal{P}}_{ijk}(0, 1) \triangleq \int_0^1 -\log\left(\left|t - \frac{1}{2}\right|\right) dt = \log(2) + 1 < \infty.$$

Instead, our Assumption IV.3 could ensure the well-definedness of $\bar{\mathcal{P}}$ as shown in the following lemma:

Lemma IV.4. *Suppose Assumption III.1, IV.1, IV.3, and Equation (7) holds, then: $\bar{\mathcal{P}}_{ijk} \to \infty$.*

Proof. By Assumption III.1, we have the finite decomposition and $\bar{\mathcal{P}}_{ijk}$ is well-defined. Without loss of generality, we assume $t \in (T_0, T_1)$ so we can pick a positive ϵ_1 such that $[t-\epsilon_1, t+\epsilon_1] \subseteq [T_0, T_1]$. For any $t' \in [t-\epsilon_1, t+\epsilon_1]$, by the boundedness of t' and Lemma IV.2, we have:

$$\text{dist}\left(b_{ij}\left(t', \theta\right), o_k\right) \le \text{dist}\left(b_{ij}\left(t, \theta\right), o_k\right) + L_1|t' - t|.$$

Putting things together, we have:

$$\begin{aligned} &\bar{\mathcal{P}}_{ijk}(T_0, T_1, \theta) \ge \bar{\mathcal{P}}_{ijk}(t-\epsilon_1, t+\epsilon_1, \theta) \\ \ge &\epsilon_1 \mathcal{P}(\text{dist}\left(b_{ij}\left(t, \theta\right), o_k\right) - d_0 + L_1\epsilon_1) \ge \epsilon_1 \mathcal{P}(\epsilon_1 + L_1\epsilon_1), \end{aligned}$$

where the last inequality is due to Equation (7) and by choosing θ so that $\text{dist}\left(b_{ij}\left(t, \theta\right), o_k\right) - d_0 \le \epsilon_1$. The lemma is proved by tending ϵ_1 to zero and applying IV.3. □

C. Feasible Interior-Point Method

We now combine the above ideas to design a feasible interior point method for the SIP problem. We divide the bounded temporal domain into a disjoint set of intervals, $[0, T] = \bigcup_l [T_0^l, T_1^l]$, and choose the midpoint constraint as the representative. As a result, the penalty functions transform the inequality-constrained NLP into an unconstrained one as follows:

$$\begin{aligned} \underset{\theta}{\text{argmin}}\ \mathcal{E}(\theta) &\triangleq \mathcal{O}(\theta) + \mu \sum_{ijkl} (T_1^l - T_0^l) \mathcal{P}_{ijkl}(\theta) \\ \mathcal{P}_{ijkl}(\theta) &\triangleq \mathcal{P}_{ijk}(\frac{T_0^l + T_1^l}{2}, \theta), \end{aligned} \tag{8}$$

where μ is a positive weight of the barrier coefficient. Note that we weight the penalty function $\mathcal{P}_{ijkl}$ by the time span $T_1^l - T_0^l$ in order to approximate the integral form $\bar{\mathcal{P}}_{ijkl}(\theta) \triangleq \bar{\mathcal{P}}_{ijk}(T_0^l, T_1^l, \theta)$ in the sense of Riemann sum. Standard first- and second-order algorithms can be utilized to solve Equation (8), where the search direction of a first-order method $d^{(1)}$ is:

$$d^{(1)} \triangleq -\nabla_\theta \mathcal{E},$$

and that of the second-order method is:

$$d^{(2)} \triangleq \mathcal{M}\left(\nabla_\theta^2 \mathcal{E}\right)^{-1} d^{(1)}.$$

Here $\mathcal{M}(\bullet)$ is an modulation function for a Hessian matrix such that $\underline{\beta} I \preceq \mathcal{M}(H) \preceq \bar{\beta} I$ for some positive constants $\underline{\beta}$ and $\bar{\beta}$. After a search direction is computed, a step size α is adaptively selected to ensure the first Wolfe's condition:

$$\mathcal{E}(\theta + d\alpha) \le \mathcal{E}(\theta) + c\left\langle d\alpha, \nabla_\theta \mathcal{E}\right\rangle, \tag{9}$$

where $c \in (0, 1)$ is a positive constant. It has been shown that if the smallest $z \in \mathbb{Z}^+$ is chosen such that $\alpha = 1/z$ satisfies Equation (9), then the feasible interior-point method will converge to the first-order critical point of $\mathcal{E}$ [44, Proposition 1.2.4]. Under the finite-precision arithmetic of a computer, we would terminate the loop of the θ update when $\|d^{(1)}\|_\infty \le \epsilon_d$. Furthermore, we add an outer loop to reduce the duality gap by iteratively reducing μ down to a small constant ϵ_μ. The overall interior point procedure of solving inequality-constrained NLP is summarized in Algorithm 1.

Algorithm 1: Feasible Interior Point Method

Input: Feasible θ, initial μ, ϵ_α, ϵ_d, ϵ_μ, $\gamma \in (0,1)$
Output: Locally optimal θ

while $\mu > \epsilon_\mu$ **do**
  $d \leftarrow d^{(1)}$ or $d \leftarrow d^{(2)}$
  while $\|d\|_\infty > \epsilon_d$ **do**
    $\alpha, \epsilon_\alpha \leftarrow$Line-Search$(\theta, d, \epsilon_\alpha)$
    $\theta \leftarrow \theta + d\alpha$
    $d \leftarrow d^{(1)}$ or $d \leftarrow d^{(2)}$
  $\mu \leftarrow \mu\gamma$
Return θ

D. Adaptive Subdivision

Algorithm 1 is used to solve NLP instead of SIP. As analyzed in Section IV-A, the feasible domain of NLP derived by surrogate constraints can be larger than that of SIP. To ensure feasibility in terms of semi-infinite constraints, we utilize the motion bound Lemma IV.2 and add an additional safety check in the line search procedure as summarized in Algorithm 2. Algorithm 2 uses a more conservative feasibility condition that shrinks the feasible domain by $\psi(T_1^l - T_0^l)$. The motion bound Lemma IV.2 immediately indicates that $\psi(x) = L_1 x/2$. However, our theoretical analysis requires an even more conservative ψ defined as:

$$\psi(x) = L_1 x/2 + L_2 x^\eta, \tag{10}$$

where L_2 and η are positive constants. We choose to only accept α found by the line search algorithm when $\theta + d\alpha$ passes the safety check. On the other hand, the failure of a safety check indicates that the surrogate constraint is not a sufficiently accurate approximation of the semi-infinite constraints and a subdivision is needed. We thus adopt a midpoint subdivision, dividing $[T_0^l, T_1^l]$ into two pieces $[T_0^l, (T_0^l+T_1^l)/2]$ and $[(T_0^l+T_1^l)/2, T_1^l]$. This procedure is repeated until α found by the line search algorithm passes the safety check. Note that the failure of safety check can be due to two different reasons: 1) the step size α is too large; 2) more subdivisions are needed. Since the first reason is easier to check and fix, so we choose to always reduce α when safety check fails, until some lower bound of α is reached. We maintain such a lower bound denoted as ϵ_α. Our line-search method is summarized in Algorithm 3. Our SIP solver is complete by combining Algorithm 1, 2, and 3.

Algorithm 2: Safety-Check(θ)

Output: $\langle i,j,k,l\rangle$ such that $\mathcal{P}_{ijkl}$ violates safety condition

for Each penalty term $\mathcal{P}_{ijkl}$ **do**
  if $\text{dist}\left(b_{ij}\left(\frac{T_0^l+T_1^l}{2}, \theta\right), o_k\right) \le d_0 + \psi(T_1^l - T_0^l)$ **then**
    Return $\langle i,j,k,l\rangle$
Return None

V. Convergence Analysis

In this section, we argue that our Algorithm 1 is suited for solving SIP problems Equation (1) by establishing three properties. First, the following result is straightforward and shows that our algorithm generates feasible iterations:

Algorithm 3: Line-Search$(\theta, d, \epsilon_\alpha)$

Input: Initial α_0, $\gamma \in (0,1)$
Output: Step size α and updated ϵ_α

$\alpha \leftarrow \alpha_0$
$\theta' \leftarrow \theta + d\alpha$
$\langle i,j,k,l\rangle \leftarrow$Safe-Check$(\theta')$
while $\langle i,j,k,l\rangle \neq$None $\vee$ θ' violates Equation (9) **do**
  if $\langle i,j,k,l\rangle \neq$None **then**
    if $\alpha \le \epsilon_\alpha$ **then**
      $\epsilon_\alpha \leftarrow \gamma\epsilon_\alpha$
      Subdivide$(\mathcal{P}_{ijkl})$ and re-evaluate $\mathcal{E}(\theta)$
      $d \leftarrow d^{(1)}$ or $d \leftarrow d^{(2)}$
    else
      $\alpha \leftarrow \gamma\alpha$
  else
    $\alpha \leftarrow \gamma\alpha$
  $\theta' \leftarrow \theta + d\alpha$
  $\langle i,j,k,l\rangle \leftarrow$Safe-Check$(\theta')$
Return α, ϵ_α

Theorem V.1. *Under Assumption III.1 and Assumption IV.1, if Algorithm 1 can find a positive α and update θ in Line 5 during an iteration, then the updated θ is a feasible solution to Equation (1).*

Proof. A step size generated by Algorithm 3 must pass the safety check, which in turn ensures that:

$$\begin{aligned}&\text{dist}\left(b_{ij}\left(\frac{T_0^l+T_1^l}{2}, \theta\right), o_k\right)\\ \ge& d_0 + \psi(T_1 - T_0) > d_0 + L_1\frac{T_1^l - T_0^l}{2},\end{aligned}$$

where we have used our choice of ψ in Equation (10). From Lemma IV.2, we have for any $t \in [T_0^l, T_1^l]$ that:

$$\begin{aligned}&\text{dist}\left(b_{ij}\left(t, \theta\right), o_k\right)\\ \ge&\text{dist}\left(b_{ij}\left(\frac{T_0^l+T_1^l}{2}, \theta\right), o_k\right) - L_1\left|t - \frac{T_0^l+T_1^l}{2}\right| > d_0.\end{aligned}$$

Since Algorithm 3 would check every spatial-temporal constraint subset, the proof is complete. □

Theorem V.1 depends on the fact that Algorithm 1 does generate an iteration after a finite amount of computation. However, the finite termination of Algorithm 1 is not obvious for two reasons. First, the line search Algorithm 3 can get stuck in the while loop and never pass the safety check. Second, even if the line search algorithm always terminate finitely, the inner while loop in Algorithm 1 can get stuck forever. This is because a subdivision would remove one and contribute two more penalty terms of form: $(T_1^l - T_0^l)\mathcal{P}_{ijkl}$ to $\mathcal{E}(\theta)$, which changes the landscape of objective function. As a result, it is possible for a subdivision to increase $\|d\|_\infty$ and Algorithm 1 can never bring $\|d\|_\infty$ down to user-specified ϵ_d. However, the

following result shows that neither of these two cases would happen by a proper choice of η:

Theorem V.2. *Under Assumption III.1, IV.1, IV.3, and suppose $\eta < 1/6$, Algorithm 1 terminates after a finite number of subdivisions.*

Proof. See Section IX. □

Theorem V.2 shows the well-definedness of Algorithm 1, which aims at solving the NLP Equation (8) instead of the original Equation (1). Our final result bridges the gap by showing that the first-order optimality condition of Equation (8) approaches that of Equation (1) by a sufficiently small choice of μ and ϵ_μ:

Theorem V.3. *We take Assumption III.1, IV.1, IV.3, X.2 and suppose $\eta < 1/6$. If we run Algorithm 1 for infinite number of iterations using null sequences $\{\mu^k\}$ and $\{\epsilon_d^k\}$, where k is the iteration number, then we get a solution sequence $\{\theta^k\}$ such that every accumulation point θ^0 satisfies the first-order optimality condition of Equation (1).*

Proof. See Section X. □

VI. Realization on Articulated Robots

We introduce two versions of our method. In our first version, we assume both the robot and the environmental geometries are discretized using triangular meshes. Although triangular meshes can represent arbitrary concave shapes, they requires a large number of elements leading to prohibitive overhead even using the acceleration techniques introduced in Section VI-C. Therefore, our second version reduces the number of geometric primitives by approximating each robot link and obstacle with a single convex hull [14, 46] or multiple convex hulls via a convex decomposition [47]. In other words, the b_{ij} and o_k in our method can be a moving point, edge, triangle, or general convex hull. In our first version, we need to ensure the two triangle meshes are collision-free. To this end, it suffices to ensure the distances between every pair of edges and every pair of vertex and triangle are larger than d_0 [48]. In our second version, we need to ensure the distance between every convex-convex pair is larger than d_0. However, it is known that edge-edge or convex-convex distance functions are not differentiable. We follow [42] to resolve this problem by bulging each edge or convex hull using curved surfaces, making them strictly convex with well-defined derivatives. In this section, we present technical details for a practical realization of our method to generate trajectories of articulated robots with translational and hinge rotational joints.

A. Computing Lipschitz Upper Bound

Our method requires the Lipschitz constant L_1 to be evaluated for each type of geometric shape. We denote by L_1^{ijk} as the Lipschitz constant for the pair of b_{ij} and o_k satisfying Lemma IV.2. We first consider the case with b_{ij} being a moving point, and all other cases are covered by minor modifications. We denote Θ as the vector of joint parameters, which is also a function of t and θ. By the chain rule, we have:

$$\begin{aligned} L_1^{ijk} &= \max \left| \frac{\partial \text{dist}(b_{ij}(t,\theta), o_k)}{\partial t} \right| \\ &= \max \left| \frac{\partial \text{dist}(b_{ij}(t,\theta), o_k)}{\partial b_{ij}(t,\theta)} \frac{\partial b_{ij}(t,\theta)}{\partial \Theta(t,\theta)} \frac{\partial \Theta(t,\theta)}{\partial t} \right|. \end{aligned}$$

Without a loss of generality, we assume each entry of $\partial\Theta(t,\theta)/\partial t$ is limited to the range $[-1,1]$. The first term is a distance function, whose subgradient has a norm at most 1 [49]. These results combined, we derive the following upper bound for L_1^{ijk} independent of o^k, which in turn is denoted as L_1^{ij}:

$$L_1^{ij} \le \max \left| \frac{\partial b_{ij}(t,\theta)}{\partial \Theta(t,\theta)} \right|_{2,1},$$

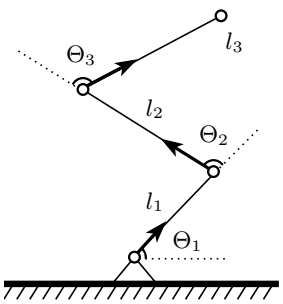
which means L_1^{ij} is upper bound of the $l_{2,1}$-norm of the Jacobian matrix (the sum of l_2-norm of each column). Next, we consider the general case with b_{ij} being a convex hull with N vertices denoted as $b_{ij}^{1,\cdots,N}$. We have the closest point on b_{ij} to o_k lies on some interpolated point $\sum_{m=1}^N b_{ij}^m \xi^m$, where ξ^m are convex-interpolation weights that are also function of $\theta(t)$. We have the following upper-bound for the distance variation over time:

$$\begin{aligned} &\left| \text{dist}\Big(\sum_{m=1}^N b_{ij}^m(t_1,\theta)\xi_1^m, o_k\Big) - \text{dist}\Big(\sum_{m=1}^N b_{ij}^m(t_2,\theta)\xi_2^m, o_k\Big) \right| \\ \le &\left| \text{dist}\Big(\sum_{m=1}^N b_{ij}^m(t_1,\theta)\xi_1^m, o_k\Big) - \text{dist}\Big(\sum_{m=1}^N b_{ij}^m(t_2,\theta)\xi_1^m, o_k\Big) \right| \\ \le &L_1^{ij}(\xi_1^m)|t_1 - t_2|. \end{aligned}$$

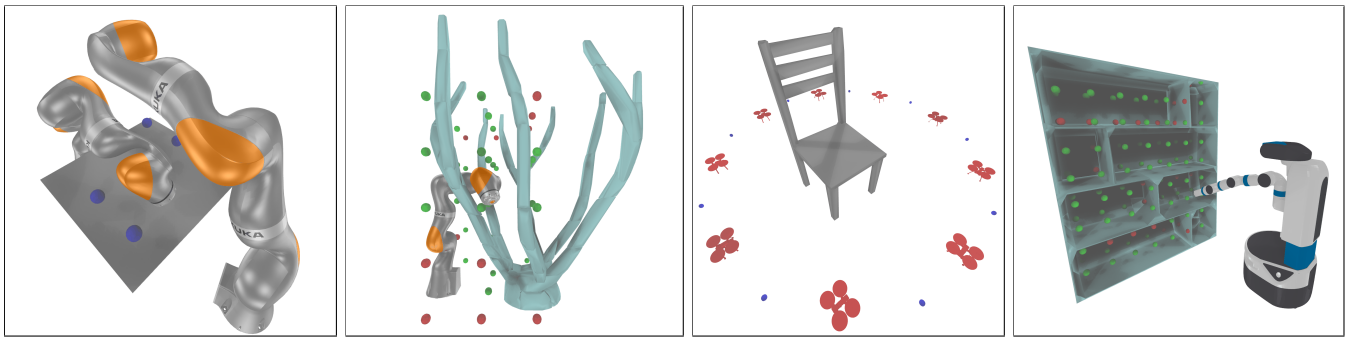
The inequality above is due to the fact that coefficients ξ^m minimize the distance, so replacing ξ_2^m with ξ_1^m will only increase the distance. Here we abbreviate $\xi_\bullet^m \triangleq \xi^m(\theta(t_\bullet))$ and assume that $\text{dist}(b_{ij}(t_1,\theta), o_k) < \text{dist}(b_{ij}(t_2,\theta), o_k)$, and we can switch t_1 and t_2 otherwise. Next, we treat ξ_1^m as a constant independent of $\theta(t)$ and estimate the ξ_1^m-dependent Lipschitz constant $L_1^{ij}(\xi_1^m)$ as:

$$\begin{aligned} L_1^{ij}(\xi_1^m) &= \max \left| \frac{\partial \text{dist}(b_{ij}(t,\theta), o_k)}{\partial \sum_{m=1}^N b_{ij}^m \xi_1^m} \frac{\partial \sum_{m=1}^N b_{ij}^m \xi_1^m}{\partial \Theta(t,\theta)} \frac{\partial \Theta(t,\theta)}{\partial t} \right| \\ &\le \max \sum_{m=1}^N \left| \frac{\partial b_{ij}^m(t,\theta)}{\partial \Theta(t,\theta)} \right|_{2,1} \xi_1^m \le \max_{m=1,\cdots,N} \left| \frac{\partial b_{ij}^m(t,\theta)}{\partial \Theta(t,\theta)} \right|_{2,1}. \end{aligned}$$

where the last inequality is due to the fact that ξ_1^m form a convex combination. We see that our estimate of $L_1^{ij}(\xi_1^m)$ is indeed independent of ξ_1^m and can be re-defined as our desired Lipschitz constant L_1^{ij}. In other words, the Lipschitz constant of a moving convex hull is the maximal Lipschitz constant over its vertices, and the Lipschitz constants of edge and triangle are just special cases of a convex hull.



It remains to evaluate the upper bound of the $l_{2,1}$-norm of the Jacobian matrix for a moving point b_{ij}. We can derive this



(a) dimension=12 (b) dimension=6 (c) dimension=54 (d) dimension=11

Fig. 3: Snapshots of our four benchmark problems with labeled dimension of configuration spaces.

bound from the forward kinematic function. For simplicity, we assume a robot arm with only hinge joints as illustrated in the inset. We use Θ_k to denote the angle of the kth hinge joint. If b_{ij} lies on the Kth link, then only $\Theta_{1,\cdots,K}$ can affect the position of b_{ij}. We assume the kth link has length l_k, then the maximal influence of Θ_k on b_{ij} happens when all the $k,\cdots,K$th links are straight, so that:

$$\left|\frac{\partial b_{ij}}{\partial \Theta_k}\right| \leq \sum_{m=1}^{k} l_m \Longrightarrow \left|\frac{\partial b_{ij}}{\partial \Theta}\right|_{2,1} \leq \sum_{k=1}^{K}\sum_{m=k}^{K} l_m.$$

B. High-Order Polynomial Trajectory Parameterization

Our L_1^{ij} formulation relies on the boundedness of $\partial\Theta(t,\theta)/\partial t$. And articulated robots can have joint limits which must be satisfied at any $t \in [0,T]$. To these ends, we use high-order composite Bézier curves to parameterize the trajectory $\Theta(t,\theta)$ in the configuration space, so that $\Theta(t,\theta)$ is a high-order polynomial function. In this form, bounds on $\Theta(t,\theta)$ at an arbitrary t can be transformed into bounds on its control points [50]. We denote the lower- and upper-joint limits as $\underline{\Theta}$ and $\bar{\Theta}$, respectively. If we denote by M_k the matrix extracting the control points of Θ_k and M_{ik} the ith row of M_k, then the joint limit constraints can be conservatively enforced by the following barrier function:

$$\sum_i \sum_k \mathcal{P}(\bar{\Theta}_k - M_{ik}\Theta_k) + \mathcal{P}(-\underline{\Theta}_k + M_{ik}\Theta_k). \tag{11}$$

A similar approach can be used to bound $\partial\Theta_k(t,\theta)/\partial t$ to the range $[-1,1]$. We know that the gradient of a Bézier curves is another Bézier curves with a lower-order, so we can denote by M_k' the matrix extracting the control points of $\partial\Theta_k(t,\theta)/\partial t$ and M_{ik}' the ith row of M_k'. The boundedness of $\partial\Theta(t,\theta)/\partial t$ for any t can then be realized by adding the following barrier function:

$$\sum_i \sum_k \mathcal{P}(1 - M_{ik}'\Theta_k) + \mathcal{P}(1 + M_{ik}'\Theta_k). \tag{12}$$

Note that these constraints are strictly conservative. However, one can always use more control points in the Bézier curve composition to allow an arbitrarily long trajectory of complex motions.

C. Accelerated & Adaptive Computation of Barrier Functions

A naive method for computing the barrier function terms $\sum_{ijkl}\mathcal{P}_{ijkl}$ could be prohibitively costly, and we propose several acceleration techniques that is compatible with our theoretical analysis. Note first that our theoretical results assume the same L_1 for all b_{ij}, but the L_1^{ij} constant computed in Section VI-A is different for each b_{ij}. Instead of letting $L_1 = \max_{ij} L_1^{ij}$, we could use a different $\phi(x) = L_1^{ij}x/2 + L_2x^\eta$ for each b_{ij}, leading to a loose safety condition and less subdivisions. Further, note that our potential function $\mathcal{P}$ is locally supported by design and we only need to compute $\mathcal{P}_{ijk}$ if the distance between b_{ij} and o_k is less than x_0+d_0. We propose to build a spatial-temporal, binary-tree-based bounding volume hierarchy (BVH) [51] for pruning unnecessary $\mathcal{P}_{ijk}$ terms, where each leaf node of our BVH indicates a unique tuple $< b_{ij}, T_0^l, T_1^l >$, which can be checked against each obstacle o_k to quickly prune $< b_{ij}, o_k >$ pairs with distance larger than x_0+d_0. The BVH further provides a convenient data-structure to perform adaptive subdivision. When safety check fails for the term $\mathcal{P}_{ijk}$, we only subdivide that single term without modifying the subdivision status of other b_{ij} and o_k pairs. Specifically, a leaf node tuple $< b_{ij}, T_0^l, T_1^l >$ is replaced by an internal node with two children: $< b_{ij}, T_0^l, (T_0^l + T_1^l)/2 >$ and $< b_{ij}, (T_0^l + T_1^l)/2, T_1^l >$ upon subdivision.

D. Handling Self-Collisions

Our method inherently applies to handle self-collisions. Indeed, for two articulated robot subsets b_{ij} and $b_{i'j'}$, we have the following generalized motion bound:

$$\begin{aligned}
&|\text{dist}(b_{ij}(t_1,\theta), b_{i'j'}(t_1,\theta)) - \text{dist}(b_{ij}(t_2,\theta), b_{i'j'}(t_2,\theta))| \\
\leq &|\text{dist}(b_{ij}(t_1,\theta), b_{i'j'}(t_1,\theta)) - \text{dist}(b_{ij}(t_2,\theta), b_{i'j'}(t_1,\theta))| + \\
&|\text{dist}(b_{ij}(t_2,\theta), b_{i'j'}(t_1,\theta)) - \text{dist}(b_{ij}(t_2,\theta), b_{i'j'}(t_2,\theta))| \\
\leq &(L_1^{ij} + L_1^{i'j'})|t_1 - t_2|,
\end{aligned}$$

where the second inequality is derived by treating $b_{i'j'}(t_1,\theta)$ and $b_{ij}(t_2,\theta)$ as a static obstacle in the first and second term, respectively. The above result implies that if Lemma IV.2 holds for distances to static obstacles, it also holds for distances between two moving robot subsets by summing up the Lipschitz constants. As a result, we can use the following alternative

definition of $\phi(x)$ within the safety check to prevent self-collisions:

$$\phi(x) = (L_1^{ij} + L_1^{i'j'})x/2 + L_2 x^{\eta}.$$

Readers can verify that all our theoretical results follow for self-collisions by the same argument, and we omit their repetitive derivations for brevity. Notably, using our adaptive subdivision scheme introduced in Section VI-C, the subdivision status of b_{ij} and $b_{i'j'}$ can be different. For example, b_{ij} can have a subdivision interval $[T_0^l, T_1^l]$, while $b_{i'j'}$ has an overlapping interval $[T_0^{l'}, T_1^{l'}]$ such that $[T_0^l, T_1^l) \cap [T_0^{l'}, T_1^{l'}) \neq \varnothing$, but $[T_0^l, T_1^l] \neq [T_0^{l'}, T_1^{l'}]$. Since we use the midpoint constraint as the representative of the interval, there is no well-defined midpoint for such inconsistent interval pairs. To tackle this issue, we note that by the midpoint subdivision rule, we have either $[T_0^l, T_1^l] \subset [T_0^{l'}, T_1^{l'}]$ or $[T_0^{l'}, T_1^{l'}] \subset [T_0^l, T_1^l]$. As a result, we could recursively subdivide the larger interval until the two intervals are identical.

VII. Evaluation

We implement our method using C++ and evaluate the performance on a single desktop machine with one 32-core AMD 3970X CPU. We make full use of the CPU cores to parallelize the BVH collision check, energy function, and derivative computations. For all our experiments, we use $x_0 = 10^{-3}$, $L_2 = 10^{-4}$, $\eta = 1/7$, $\mu = 10^{-2}$, $\epsilon_d = 10^{-4}$. The trajectory is parameterized in configuration space using a 5th-order composite Bézier curve with 5 segments over a horizon of $T = 5$s. We use four computational benchmarks discussed below. For all these benchmarks, we use trivial initialization, where we pick collision-free robot poses and have the robots stand still throughout the trajectory. Our goal is to have these robots reach one or more goal configurations as specified by objective functions.

A. Benchmark Problems

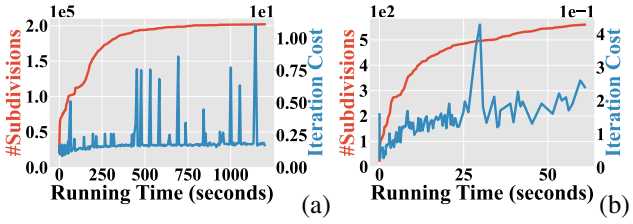


Fig. 4: The number of subdivisions and cost per iteration in seconds plotted against the computational time for our first benchmark using triangular mesh (a) and convex hull (b) representations.

Our first benchmark (Figure 3a) involves two LBR iiwa robot arms simultaneously reaching 4 target points in a shared workspace. To this end, our objective function involves a distance measure between the robot end-effectors and the target points, as well as a Laplacian trajectory smoothness metric as in [22]. The convergence history as well as the number of subdivisions is plotted in Figure 4 for both versions of geometric representations: triangular mesh and convex hull. Our method using triangular meshes is much slower than that using convex hulls, due to the fine geometric details leading to a large number of triangle-triangle pairs. Our method with the convex hull representation converges after 359 subdivisions, 231 iterations, and 2.23 minutes of computation to reach all 4 target positions. We have also plotted the cost per iteration in Figure 4, which increases as more subdivisions and barrier penalty terms are introduced.

Our second benchmark (Figure 3b with successful points in green and unsuccessful points in red) involves a single LBR iiwa robot arm interacting with a tree-like obstacle with thin geometric objects. Such obstacles can lead to ill-defined gradients for infeasible discretization methods or even tunnel through robot links [1], while our method can readily handle such ill-shaped obstacles. We sample a grid of target positions for the end-effector to reach and run our algorithm for each position. Our method can successfully reach 16/56 positions. We further conduct an exhaustive search for each unsuccessful point. Specifically, if an unsuccessful point is neighboring a successful point, we optimize an additional trajectory for the end-effector to connect the two points. Such connection is performed until no new points can be reached. In this way, our method can successfully reach 39/56 positions. On average, to reach each target point, our method with the convex hull representation converges after 416 subdivisions, 246 iterations, and 0.31 minutes of computation, so the total computational time is 17.36 minutes.

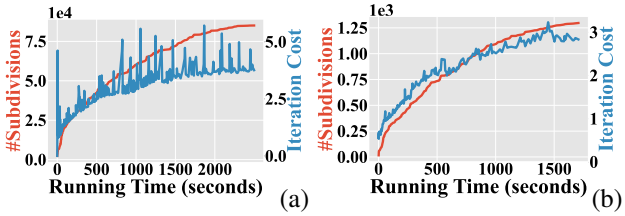


Fig. 5: The number of subdivisions and cost per iteration in seconds plotted against the computational time for our third benchmark using triangular mesh (a) and convex hull (b) representations.

Our third benchmark (Figure 3c) involves a swarm of UAVs navigating across each other through an obstacle. A UAV can be modeled as a free-flying rigid body with 6 degrees of freedom. Prior work [52] searches for only the translations and then exploits differential flatness [53] to recover feasible orientation trajectories. However, such orientation trajectories might not be collision-free. Instead, we can optimize both the translation and rotation with a collision-free guarantee. We have experimented with both geometric representations, and the convergence history of this example is plotted Figure 5. Again, our method with triangular mesh can represent detailed geometries but takes significantly more computation time. In comparison, our method with the convex hull representation converges after 1297 subdivisions, 188 iterations, and 25.87 minutes of computation. In Table I, we summarize the fraction of computation for each component of our method for both of the triangle mesh and convex hull representations. Our major bottleneck lies in the energy evaluation, i.e. computing $\mathcal{E}(\theta)$

Step	Time
Line-search direction computation	1.6%
Subdivision	3.3%
Safety-check	14.4%
Objective function evaluation	80.7%

TABLE I: Over an optimization, we summarize the fraction of computation on each component. Our major bottleneck lies in the energy evaluation, taking 80.7% of the computation.

and its derivatives. Note that we use either triangular mesh or convex hull as our geometric representation. In the former case, each distance term is between a pair of edges or a pair of point and triangle. Although such distance function is cheap to compute, the number of $\mathcal{P}_{ijkl}$ terms is large, leading to a major computational burden. In the latter case, the number of $\mathcal{P}_{ijkl}$ terms is much smaller, but each evaluation of $\mathcal{P}_{ijkl}$ involves the non-trivial computation of the shortest distance between two general convex hulls, which is also the computational bottleneck.

Our final benchmark (Figure 3d) plans for an armed mobile robot to reach a grid of locations on a book-shelf. We use convex decomposition to represent both the robot links and the book-shelf as convex hulls. Starting from a faraway initial guess, our method can guide the robot to reach 88/114 target positions, achieving a success rate of 77.19%. On average over each target point, our method with the convex hull representation converges after 874 subdivision, 402 iterations, and 1.77 minutes of computation, so the total computational time is 201.78 minutes.

ith Link	1	2	3	4	5	6	7	9	10
δL_1^i	3.71	5.24	4.87	5.06	5.84	6.14	6.88	7.21	7.79

TABLE II: Overestimation of Lipschitz constant for each link of the armed mobile robot in Figure 3d (The 1st link is closest to the root joint).

B. Conservativity of Lipschitz Constant

According to Table I, our main computational bottleneck lies in the large number of penalty terms resulting from repeated subdivision. This is partly due to over-estimation of the Lipschitz constant, resulting in an overly conservative motion bound and more subdivisions to tighten the bound. To quantify the over-estimation for the ith rigid body, we compare our Lipschitz constant with the following groundtruth tightest bound:

$$L_1^{ij*} \triangleq \underset{t\in[T_0^l,T_1^l]}{\operatorname{argmin}} \left|\frac{\partial b_{ij}(t,\theta)}{\partial t}\right|,$$

and define the over-estimation metric as $\delta L_1^i \triangleq \max_j L_1^{ij}/L_1^{ij*}$, where we calculate L_1^{ij*} by sampling time instances within $t \in [T_0^l, T_1^l]$ at an interval of $\delta t = 10^{-3}$ and pick the largest value. We initialize 1000 random trajectories lasting for 1s ($T_1^l - T_0^l = 1$) for the armed mobile robot in Figure 3d and we summarize the average ΔL_1^i over 1000 cases for each robot link in Table II, with the 1st link being closest to the root joint. The level of over-estimation increases for links further down the kinematic chain, due to the over-estimation of each joint on the chain. As a result, more subdivisions are needed with longer kinematic chains.

C. Complexity of Geometric Representation

Apart from the over-estimation of the Lipschitz constant, the inherent complexity of the robot geometry contributes majorly to the number of penalty terms, especially when triangular meshes are used. In Figure 6, we analyze the computational time increase with respect to the number of triangles. We use our first benchmark (Figure 3a) and subdivide the triangular meshes uesd to represent the two robot arms. Each subdivision increases the number of triangles by a factor of 4. Finally, we plot the total computational time as well as the average time per iteration. The result shows that computational time grows nearly linearly with the number of triangles.

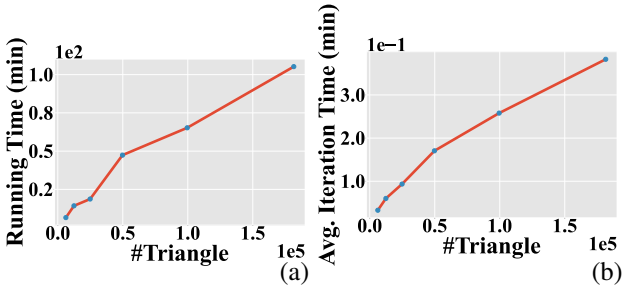


Fig. 6: In our first benchmark (Figure 3a), we plot the total computational time (a) and the average time per iteration (b) against the number of triangles used to represent the robot links.

D. Comparisons with Exchange Method

We combine the merits of prior works into a reliable implementation of the exchange method [12]. During each iteration, the spatio-temporal deepest penetration point between each pair of convex objects is detected and inserted into the finite index set. To detect the deepest penetration point, we densely sample the temporal domain with a finite interval of $\epsilon = 10^{-3}$ and compute penetration depth for every pair of robot links and each time instance. We use the exact point-to-mesh distance computation implemented in CGAL [54] to compute penetration depth at each time instance, and we adopt the bounding volume hierarchy and branch-and-bound technique to efficiently prune non-deepest penetrated points as proposed in [6]. After the index set is updated, an NLP is formulated and solved using the IPOPT software package [10]. We terminate NLP when the inf-norm of the gradient is less than 10^{-4}, same as for our method. Note that we sample the temporal domain with a finite interval of $\epsilon = 10^{-3}$ and we only insert new constraints into the index set and never remove constraints from it. Therefore, our exchange solver is essentially reducing SIP to NLP solver with progressive constraint instantiation, so that our solver pertains to the same feasibility guarantee as an NLP solver. Such is the strongest feasibility guarantee an exchange-based SIP solver can provide, to the best of our knowledge. For

fairness of comparison, we only use the collision constraints for both methods, i.e. the joint limit Equation (11) and gradient bound Equation (12) are not used in our method for this section.

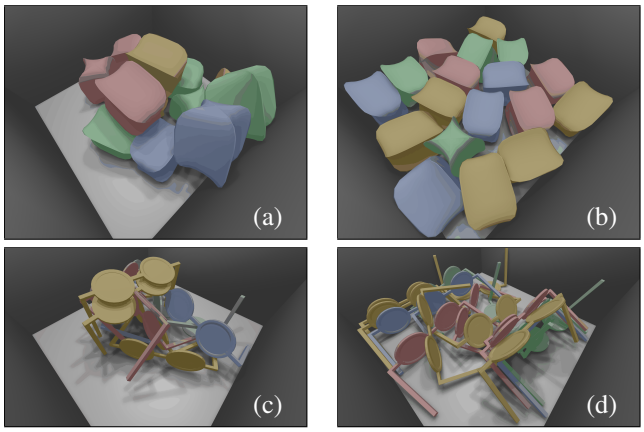


Fig. 7: The result of object settling for 18 bulky objects one-by-one (a) and jointly (b) using our method. Further, we can settle very thin objects such as eyeglasses, either one-by-one (c) or jointly (d).

#Bulky Object	1	2	3	4	5	6	7	9	10
Discretization	2.13	4.31	6.67	5.96	9.33	6.18	12.44	21.54	14.43
Exchange	0.11	0.06	0.11	0.14	0.22	0.31	0.84	1.93	0.40
#Thin Object	1	2	3	4	5	6	7	9	10
Discretization	3.90	8.43	4.74	22.87	24.10	9.71	23.43	10.87	11.94
Exchange	-	-	-	-	-	-	-	-	-

TABLE III: Computational time in seconds for settling 10 objects using our discretization method or the exchange method. We use "-" to indicate failed runs.

Our first comparative benchmark involves object settling, where we drop non-convex objects into a box and use both methods to compute the force equilibrium poses. This can be achieved by setting the objective function to the gravitational potential energy and optimizing a single pose for all the objects. As compared with trajectory optimization, object settling is a much easier task, since no temporal subdivision or sampling is needed. We consider two modes of object settling: the first one-by-one mode sequentially optimizes one object's pose per-run, assuming all previously optimized objects stay still; the second joint mode optimizes all the objects' poses in a single run. The one-by-one mode is faster to compute, but cannot find accurate force equilibrium poses. The joint mode can find accurate poses, but takes considerably more iterations to converge. In our first experiment, we drop 18 bulky objects into a box and the results are shown in Figure 7a and the average computational time for the first 10 objects is summarized in Table III. The exchange method is more than 10× faster than our method. This is because our method needs to consider all potential contacts between all pairs of triangles during each iteration. In our second experiment, we drop 18 eyeglasses into the box as illustrated in Figure 7c. Our method can still find force equilibrium poses, but the exchange method fails to find a feasible solution, because the eyeglasses have thin geometries with no well-defined penetration depth. Finally, we run our method in joint mode for both examples, where our method takes 19 minutes for the bulky objects and 20 minutes for the thin objects. As illustrated in Figure 7bd, the joint mode computes poses with much lower gravitational potential energy.

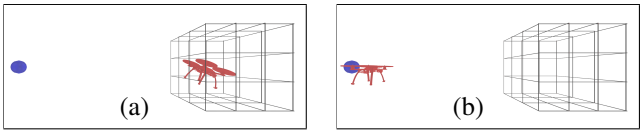


Fig. 8: A UAV is trapped inside a cage but trying to fly outside to the blue target point. We show final location of the UAV computed by our discretization method (a) and the exchange method (b).

Our second comparative benchmark uses a toy example illustrated in Figure 8, where a UAV is trapped in a cage consisting of a row of metal bars. The UAV is trying to fly to the target point outside. Our method would get the UAV close to the target point but still trapped inside the cage, while the exchange method will erroneously get the UAV outside the cage however small ϵ is. This seemingly surprising result is due to the fact that the exchange method ultimately detects penetrations at discrete time instances with a resolution of ϵ. However, the optimizer is allowed to arbitrarily accelerate the UAV to a point where it can fly out of the cage within ϵ, and the collision constraint will be missed.

Finally, we run the two algorithms on the first and third benchmarks (Figure 3ac), which involve only a single target position for the robots. (The other two benchmarks involve a grid of target positions making it too costly to compute using the exchange method.) The exchange method succeeds in both benchmarks. On the first benchmark, the exchange method takes 33.09 minutes to finish the computation after 123 index set updates and NLP solves, while our method only takes 5.47 minutes. Similarly, on the third benchmark, the exchange method takes 699.23 minutes to finish the computation after 1804 index set updates and NLP solves, while our method only takes 30.02 minutes. The performance advantage of our method is for two reasons: First, our method does not require the collision to be detected at the finest resolution ($\epsilon = 10^{-3}$ in the exchange method). Instead, we only need a subdivision that is sufficient to find descendent directions. Second, our method does not require the NLP to be solved exactly after each update of the energy function. We only run one iteration of Newton's method per round of subdivision.

E. Comparison with Sampling-Based Method

In contrast to our locally optimal guarantee, sampling-based methods provide a stronger asymptotic global optimality guarantee, so we use the open-source sampling-based algorithm implementation [55] as a groundtruth and set their low-level discrete collision checker to use a small sampling interval of $\epsilon = 10^{-3}$. In our first comparison, we run our method and RRT-Star on the first benchmark (Figure 3a) with a

timeout of 60 minutes. For both methods, we set the objective function to be the weighted combination of the configuration-space trajectory length and the Cartesian-space distance to the target end-effector position. Throughout the optimizations, we save the best trajectory every 10 seconds and profile their trajectory length and distance to target in Figure 9. Our method converges much faster to a locally optimal solution, while RRT-Star takes a long computational time to search for better trajectories in the 12-dimensional configuration space, making little progress. We have also tested RRT-Star on the third benchmark (Figure 3c) with an even higher 54-dimensional configuration space, but it fails to compute a meaningful trajectory within 60 minutes. This is due to the exponential complexity of the zeroth-order sampling-based algorithms.

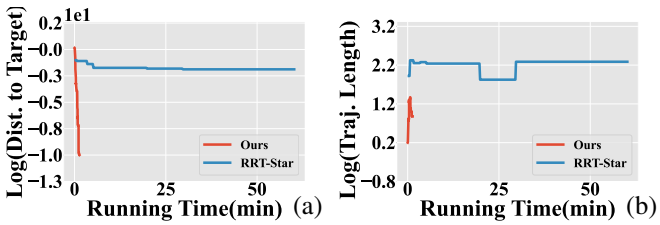


Fig. 9: We profile the Cartesian-space distance to target (a) and the configuration-space trajectory length (b) over a trajectory optimization procedure using our algorithm and RRT-Star, both in log-scale.

In our second comparison, we run PRM on the second benchmark (Figure 3b) with a timeout of 20 minutes, which is already longer than the computational time taken by our method to reach all target positions, and we consider a target position reached if the end-effector is within 7 centimeters from it. In this 6-dimensional configuration space, PRM succeeds in reaching 46/56 target positions, while our method reaches 39/56 target positions. Our lower success rate is due to our locally optimal nature. For a higher success rate, the sampling-based algorithms can be combined with our local method to provide a high-quality initial guess, e.g., as done in [56], but this is beyond the scope of this research. We further test PRM on the fourth benchmark (Figure 3d) with a timeout of 240 minutes, which is again longer than the total computational time of our method. In this 11-dimensional configuration space, PRM can only reach 17/114 target positions, while our method reaches 88/114 target positions, highlighting the benefits of our first-order algorithms.

Interval ϵ	0.1	0.05	0.025	0.0125	0.01
#UAV-Escape	10/10	7/10	2/10	0/10	0/10

TABLE IV: The behavior of UAV planned by RRT-Star under different sample intervals ϵ.

Finally, we compare the two methods on the toy example in Figure 8 to highlight the sensitivity of RRT-Star to the sampling interval of the collision checker. Again, we set the timeout to 30 minutes for RRT-Star. We use different sampling interval ϵ and run RRT-Star for 10 times under each ϵ. The number of times when UAV can incorrectly escape the cage is summarized in Table IV. The UAV is correctly trapped in the cage under sufficiently small ϵ, while it escapes under large ϵ. For intermediary values of ϵ, the behavior of UAV depends on the random sample locations. In contrast, our method can guarantee that the UAV is correctly trapped inside the cage without tuning ϵ.

F. Comparison with STOMP

We conduct additional comparison with STOMP [57], a widely used gradient-guided trajectory optimizer. This method uses stochastic approximation to compute gradient for a general, non-smooth cost function. As a result, it is convenient for us to formulate collision-avoidance constraints as a soft penalty term. Specifically, we use the same penetration detector as in [6], which finds the deepest penetrating point between each pair of point cloud and the convex hull. The absolute penetration depth are summed up and used as an additional cost function. We use the same setting for other other cost functions and parameters and compare the performance of the two methods on our third benchmark (Figure 3c). For STOMP, we use 12 samples to approximate the gradient and the evaluation of cost functions are computed in parallel for each sample. Note that stochastic gradient approximation is slower than analytic gradients used in our method. However, this problem can be alleviated using finite-difference or approximate analytic gradient used by follow-up works [23, 58]. To factor out the influence of efficiency in gradient calculation as much as possible, we plot the iteration-wise change of smoothness cost and distance to target cost in Figure 10. The results show that, while both methods converge, our method achieves much faster reduction in both cost terms and ultimately converges to a better solution.

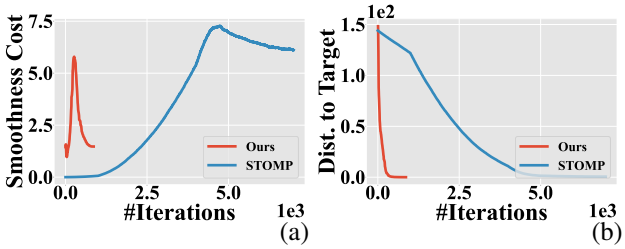


Fig. 10: On our third benchmark (Figure 3c), we plot the iteration-wise change of smoothness cost (a) and distance to target (b) for our method and STOMP.

VIII. Conclusion

We propose a provably feasible algorithm for collision-free trajectory generation of articulated robots. We formulate the underlying SIP problem and solve it using a novel feasible discretization method. Our method divides the temporal domain into discrete intervals and chooses one representative constraint for each interval, reducing the SIP to an NLP. We further propose a conservative motion bound that ensures the original SIP constraint is satisfied. Finally, we establish theoretical convergence guarantee and propose practical implementations for articulated robots. Our results show that our

method can generate long-horizon trajectories for industrial robot arms within a couple minutes of computation.

Our method pertains several limitations that we consider to address as future work. On the downside, our method requires a large number of subdivisions to approach a feasible and nearly optimal solution. This is partly due to an overly conservative Lipschitz constant estimation and a linear motion bound Lemma IV.2. In the future, we plan to reduce the number of subdivisions and improve the computational speed by exploring high-order Taylor models [29]. As a minor limitation, our method requires a customized line-search scheme and cannot use off-the-shelf NLP solvers, which potentially increase the implementation complexity. Finally, our method does not account for equality constraints, which is useful for modeling the dynamics of the robot. Additional equality constraints can be incorporated by combining our method with merit-function-based techniques [44], which is an essential avenue of future work.

REFERENCES

[1] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[2] K. J. Kyriakopoulos and G. N. Saridis, "Minimum jerk path generation," in *Proceedings. 1988 IEEE international conference on robotics and automation*, IEEE, 1988, pp. 364–369.

[3] C. Hansen, J. Öltjen, D. Meike, and T. Ortmaier, "Enhanced approach for energy-efficient trajectory generation of industrial robots," in *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, IEEE, 2012, pp. 1–7.

[4] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," *Robotics: Science and Systems VIII*, pp. 1–8, 2012.

[5] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[7] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 3859–3866.

[8] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 46–61, 2018.

[9] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *Robotics: science and systems*, Citeseer, vol. 9, 2013, pp. 1–10.

[10] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

[11] K. Hauser, "Semi-infinite programming for trajectory optimization with non-convex obstacles," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1106–1122, 2021.

[12] M. López and G. Still, "Semi-infinite programming," *European journal of operational research*, vol. 180, no. 2, pp. 491–518, 2007.

[13] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *arXiv preprint arXiv:2205.04422*, 2022.

[14] A. Amice, H. Dai, P. Werner, A. Zhang, and R. Tedrake, "Finding and optimizing certified, collision-free regions in configuration space for robot manipulators," *arXiv preprint arXiv:2205.03690*, 2022.

[15] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[16] S. J. Wright, *Primal-dual interior-point methods*. SIAM, 1997.

[17] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," in *Robotics Research: Results of the 12th International Symposium ISRR*, Springer, 2007, pp. 83–97.

[18] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," 2013.

[19] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 1478–1483.

[20] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 2997–3004.

[21] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 2951–2957.

[22] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[23] M. Zucker *et al.*, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[24] A. L. Tits, "Feasible sequential quadratic programmingfeasible sequential quadratic programming," in *Encyclopedia of Optimization*, C. A. Floudas and

P. M. Pardalos, Eds. Boston, MA: Springer US, 2009, pp. 1001–1005.

[25] Y.-K. Choi, W. Wang, Y. Liu, and M.-S. Kim, "Continuous collision detection for two moving elliptic disks," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 213–224, 2006.

[26] Y.-K. Choi, J.-W. Chang, W. Wang, M.-S. Kim, and G. Elber, "Continuous collision detection for ellipsoids," *IEEE Transactions on visualization and Computer Graphics*, vol. 15, no. 2, pp. 311–325, 2008.

[27] T. Brochu, E. Edwards, and R. Bridson, "Efficient geometrically exact continuous collision detection," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–7, 2012.

[28] F. Schwarzer, M. Saha, and J.-C. Latombe, "Exact collision checking of robot paths," *Algorithmic foundations of robotics V*, pp. 25–41, 2004.

[29] X. Zhang, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using taylor models and temporal culling," *ACM Trans. Graph.*, vol. 26, no. 3, 15–es, Jul. 2007.

[30] J. Pan, L. Zhang, and D. Manocha, "Fast smoothing of motion planning trajectories using b-splines," in *Robotics: Science and Systems*, 2011.

[31] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 4054–4061.

[32] A. Clark, "Verification and synthesis of control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*, IEEE, 2021, pp. 6105–6112.

[33] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[34] M. Zhang and K. Hauser, "Semi-infinite programming with complementarity constraints for pose optimization with pervasive contact," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 6329–6335.

[35] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 42–49.

[36] P. A. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

[37] T. Pietrzykowski, "An exact potential method for constrained maxima," *SIAM Journal on numerical analysis*, vol. 6, no. 2, pp. 299–304, 1969.

[38] A. R. Conn and N. I. Gould, "An exact penalty function for semi-infinite programming," *Mathematical Programming*, vol. 37, no. 1, pp. 19–40, 1987.

[39] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

[40] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*, IEEE, 2019, pp. 3420–3431.

[41] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2017, pp. 2872–2879.

[42] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar, "A strictly convex hull for computing proximity distances with continuous gradients," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 666–678, 2014.

[43] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun, "Asynchronous contact mechanics," in *ACM SIGGRAPH 2009 Papers*, ser. SIGGRAPH '09, New York, NY, USA: Association for Computing Machinery, 2009.

[44] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[45] U. Schättler, "An interior-point method for semi-infinite programming problems," *Annals of Operations Research*, vol. 62, no. 1, pp. 277–301, 1996.

[46] H. Dai, A. Majumdar, and R. Tedrake, "Synthesis and optimization of force closure grasps via sequential semidefinite programming," in *Robotics Research*, Springer, 2018, pp. 285–305.

[47] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polygons," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 17–26.

[48] M. Tang, D. Manocha, and R. Tong, "Fast continuous collision detection using deforming non-penetration filters," in *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, New York, NY, USA: ACM, 2010, pp. 7–13.

[49] S. Osher and R. P. Fedkiw, *Level set methods and dynamic implicit surfaces*. Springer New York, 2005, vol. 1.

[50] E. V. Shikin and A. I. Plis, *Handbook on Splines for the User*. CRC press, 1995.

[51] Y. Gu, Y. He, K. Fatahalian, and G. Blelloch, "Efficient bvh construction via approximate agglomerative clustering," in *Proceedings of the 5th High-Performance Graphics Conference*, 2013, pp. 81–88.

[52] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 5332–5339.

[53] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[54] A. Fabri and S. Pion, "Cgal: The computational geometry algorithms library," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2009, pp. 538–539.

[55] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, Dec. 2012, https://ompl.kavrakilab.org.

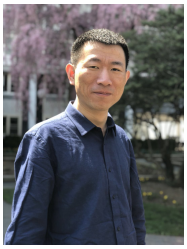
[56] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (rabit): A framework to integrate local information into optimal path planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 4207–4214.

[57] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 4569–4574.

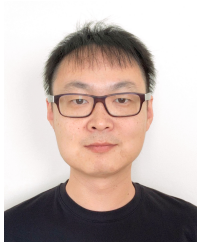
[58] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs.," in *Robotics: Science and Systems*, vol. 12, 2016.

Duo Zhang is a first-year Ph.D. student at the Algorithmic Robotics & Control Lab, Rutgers University advised by Prof. Jingjin Yu. He received his MS degree from NYU under the guidance of Prof. Daniele Panozzo and Prof. Lerrel Pinto. He used to work as an R&D intern at Light-Speed Studios, Tencent America. His research focuses on motion planning, physical simulation & manipulation, and robot learning.

Chen Liang is an incoming Ph.D. student at the Applied Planning, Learning, and Optimization (Apollo) Lab, Yale University. He received his B.E. degree in Computer Science from Zhejiang University. He used to work as a research intern at Light-Speed Studios, Tencent. His research focuses on optimization-based planning, control, and physical-based simulation.



Xifeng Gao is currently a principal researcher with Tencent North America. He has more than 10 years of academic research experience. He is interested in solving geometric computing related problems in research areas, such as Computer Graphics, Digital Games, CAD/CAE, Multimedia Processing, Robotics, and Digital Fabrication. Please refer to https://gaoxifeng.github.io/ for more details.



Kui Wu is a principal research scientist in Light-Speed Studios, Tencent America. Previously, he was a postdoctoral associate in the Computational Design & Fabrication Group under the guidance of Prof. Wojciech Matusik at MIT CSAIL. He received his Ph.D. degree in Computer Science from University of Utah in 2019, advised by Prof. Cem Yuksel. His research interests are computer graphics, especially on mesh processing, knitting, real-time rendering, and physical-based simulation.

Zherong Pan is a senior researcher at Tencent America. He was a postdoctoral researcher at the Intelligent Motion Lab of the University of Illinois Urbana-Champaign. He obtained Ph.D. degree from the University of North Carolina at Chapel Hill. His research is focused on numerical analysis, motion planning, and physics-based modeling for high-dimensional dynamic systems.

IX. Finite Termination of Algorithm 1

We show that Algorithm 1 would terminate after finitely many iterations. We will omit the parameter of a function whenever there can be no confusion. Our main idea is to compare the following two terms:

$$\tilde{\mathcal{P}} \triangleq \sum_{ijkl}(T_1^l - T_0^l)\mathcal{P}_{ijkl} \quad \bar{\mathcal{P}} \triangleq \sum_{ijkl}\int_{T_0^l}^{T_1^l}\mathcal{P}_{ijk}(t)dt,$$

where we use a shorthand notation $\tilde{\mathcal{P}}$ for the penalty function part of Equation (8). Conceptually, $\tilde{\mathcal{P}}$ approximates $\bar{\mathcal{P}}$ in the sense of Riemann sum and the approximation error would reduce as more subdivisions are performed. However, the approximation error will not approach zero because our subdivision is adaptive. Therefore, we need a new tool to analyze the difference between $\tilde{\mathcal{P}}$ and $\bar{\mathcal{P}}$. To this end, we introduce the following hybrid penalty function with a variable ϵ_2 controlling the level of hybridization:

$$\hat{\mathcal{P}}(\epsilon_2) = \sum_{ijkl}\begin{cases}(T_1^l - T_0^l)\mathcal{P}_{ijkl} & T_1^l - T_0^l \geq \epsilon_2\\ \int_{T_0^l}^{T_1^l}\mathcal{P}_{ijk}(t)dt & T_1^l - T_0^l < \epsilon_2,\end{cases}$$

where we use the integral form when a temporal interval is shorter than ϵ_2, and use the surrogate constraint otherwise. An important property of $\hat{\mathcal{P}}(\epsilon_2)$ is that it is invariant to subdivision after finitely many iterations:

Lemma IX.1. *Given fixed ϵ_2, and after finitely many times of subdivision, $\hat{\mathcal{P}}(\epsilon_2)$ becomes invariant to further subdivision.*

Proof. Since each subdivision would reduce a time interval by a factor of $1/2$, it takes finitely many subdivisions to reduce a time interval to satisfy: $T_1^l - T_0^l < \epsilon_2$. Therefore, after finitely many subdivision operators, a time interval must satisfy one of two cases: (Case I) No more subdivisions are applied to it, making it invariant to further subdivisions; (Case II) The time interval $T_1^l - T_0^l < \epsilon_2$ and infinitely many subdivisions will be applied, but the integral is invariant to subdivision. □

Next, we show that the difference between $\hat{\mathcal{P}}$ and $\tilde{\mathcal{P}}$ is controllable via ϵ_2. The following result bound their differences:

Lemma IX.2. *Taking Assumption III.1, IV.1, IV.3, and assuming θ is generated by some iteration of Algorithm 1, we have:*

$$\left|\hat{\mathcal{P}}(\epsilon_2) - \tilde{\mathcal{P}}\right| = \boldsymbol{O}(\epsilon_2^{1-5\eta}),$$

for arbitrarily small fixed ϵ_2.

Proof. We use the shorthand notation $\sum_{ijkl}^{\Delta T<\epsilon_2}$ to denote a summation over intervals $T_1^l - T_0^l < \epsilon_2$, and the following abbreviations are used:

$$d_t \triangleq \text{dist}\left(b_{ij}\left(t,\theta\right), o_k\right) - d_0$$
$$d_m \triangleq \text{dist}\left(b_{ij}\left(\frac{T_0^l + T_1^l}{2}, \theta\right), o_k\right) - d_0.$$

Since θ is generated by line search, we have θ passes the safety check, leading to the following result:

$$|\mathcal{P}_{ijk}(t) - \mathcal{P}_{ijkl}| = \left|\int_{d_t}^{d_m}\frac{d\mathcal{P}(x)}{dx}dx\right|$$
$$\leq \left|\frac{d\mathcal{P}(x)}{dx}\right|_{L_2(T_1^l - T_0^l)^\eta}\left||d_m - d_t|\right|$$
$$\leq L_1\left|\frac{d\mathcal{P}(x)}{dx}\right|_{L_2(T_1^l - T_0^l)^\eta}\left|\left|t - \frac{T_0^l + T_1^l}{2}\right|\right|.$$

The second inequality above is due to the safety check condition and monotonicity of $\mathcal{P}, |\nabla_x\mathcal{P}|$. The third inequality above is due to Lemma IV.2. The result in our lemma is derived immediately as follows:

$$\left|\hat{\mathcal{P}}(\epsilon_2) - \tilde{\mathcal{P}}\right| \leq \sum_{ijkl}^{\Delta T<\epsilon_2}\int_{T_0^l}^{T_1^l}\left|\mathcal{P}_{ijk}(t) - \mathcal{P}_{ijkl}\right|dt$$
$$\leq \sum_{ijkl}^{\Delta T<\epsilon_2} L_1\left|\frac{d\mathcal{P}(x)}{dx}\right|_{L_2(T_1^l - T_0^l)^\eta}\left|\int_{T_0^l}^{T_1^l}\left|t - \frac{T_0^l + T_1^l}{2}\right|dt\right.$$
$$\leq \sum_{ijkl}^{\Delta T<\epsilon_2} L_1\frac{(T_1^l - T_0^l)^2}{4}\left|\frac{d\mathcal{P}(x)}{dx}\right|_{L_2(T_1^l - T_0^l)^\eta}\Bigg|$$
$$\leq \sum_{ijkl}\frac{T+\epsilon_2}{\epsilon_2}L_1\frac{(T_1^l - T_0^l)^2}{4}\left|\frac{d\mathcal{P}(x)}{dx}\right|_{L_2(T_1^l - T_0^l)^\eta}\Bigg|$$
$$= \sum_{ijkl}\frac{T+\epsilon_2}{\epsilon_2}L_1(T_1^l - T_0^l)\Gamma(T_1^l - T_0^l)$$
$$\Gamma(T_1^l - T_0^l) \triangleq \frac{T_1^l - T_0^l}{4}\left|\frac{d\mathcal{P}(x)}{dx}\right|_{L_2(T_1^l - T_0^l)^\eta}\Bigg|.$$

The last inequality above is by the assumption that the entire temporal domain $[0,T]$ is subdivided into intervals of length smaller than ϵ_2. It can be shown by direct verification that by choosing $\eta < 1/5$, $\Gamma(T_1^l - T_0^l) = \mathbf{O}(\epsilon_2^{1-5\eta})$ as $T_1^l - T_0^l \to 0$ and the lemma is proved. □

In a similar fashion to Lemma IX.2, we can bound the difference in gradient:

Lemma IX.3. *Taking Assumption III.1, IV.1, IV.3, and assuming θ is generated by some iteration of Algorithm 1, we have:*

$$\left\|\nabla_\theta\hat{\mathcal{P}}(\epsilon_2) - \nabla_\theta\tilde{\mathcal{P}}\right\| = \boldsymbol{O}(\epsilon_2^{1-6\eta}),$$

for arbitrarily small fixed ϵ_2.

Proof. Again, we use the shorthand notations: $\sum_{ijkl}^{\Delta T<\epsilon_2}$, d_t, and d_m. We begin by bounding the error of the integrand:

$$\left\|\nabla_\theta\mathcal{P}_{ijk}(t) - \nabla_\theta\mathcal{P}_{ijkl}\right\|$$
$$= \left\|\frac{d\mathcal{P}(d_t)}{dd_t}\nabla_\theta d_t - \frac{d\mathcal{P}(d_m)}{dd_m}\nabla_\theta d_m\right\|$$
$$\leq \left|\frac{d\mathcal{P}(d_t)}{dd_t}\right|\left\|\nabla_\theta d_t - \nabla_\theta d_m\right\| + \left|\frac{d\mathcal{P}(d_t)}{dd_t} - \frac{d\mathcal{P}(d_m)}{dd_m}\right|\left\|\nabla_\theta d_m\right\|,$$

which is due to triangle inequality. There are two terms in the last equation to be bounded. To bound the first term, we use a similar argument as Lemma IV.2. Under Assumption IV.1, there must exist some constant L_3 such that:

$$\left\|\nabla_\theta d_t - \nabla_\theta d_m\right\| \leq L_3\left|t - \frac{T_0^l + T_1^l}{2}\right|.$$

Since θ passes the safety check, we further have:

$$\left|\frac{d\mathcal{P}(d_t)}{dd_t}\right| \|\nabla_\theta d_t - \nabla_\theta d_m\| \le L_3 \left|\frac{d\mathcal{P}(x)}{dx}\right|_{L_2(T_1^l - T_0^l)^\eta} \left| t - \frac{T_0^l + T_1^l}{2} \right|.$$

To bound the second term, we note that $\|\nabla_\theta d_m\| \le L_4$ for some L_4 because its domain is bounded. By the mean value theorem, we have:

$$\begin{aligned}
&\left|\frac{d\mathcal{P}(d_t)}{dd_t} - \frac{d\mathcal{P}(d_m)}{dd_m}\right| \|\nabla_\theta d_m\| \le L_4 \left| \int_{d_t}^{d_m} \frac{\partial^2 \mathcal{P}(x)}{\partial x^2} dx \right| \\
\le & L_4 \left|\frac{\partial^2 \mathcal{P}(x)}{\partial x^2}\right|_{L_2(T_1^l - T_0^l)^\eta} \left| t - \frac{T_0^l + T_1^l}{2} \right|,
\end{aligned}$$

where we have used the safety check condition and monotonicity of $|\nabla_x^2 \mathcal{P}|$. Putting everything together, we establish the result in our lemma as follows:

$$\begin{aligned}
&\left\|\nabla_\theta \hat{\mathcal{P}}(\epsilon_2) - \nabla_\theta \tilde{\mathcal{P}}\right\| \le \sum_{ijkl}^{\Delta T < \epsilon_2} \\
&L_3 (T_1^l - T_0^l) \Gamma(T_1^l - T_0^l) + L_4 (T_1^l - T_0^l) \Gamma'(T_1^l - T_0^l) \\
&\Gamma'(T_1^l - T_0^l) \triangleq \frac{T_1^l - T_0^l}{4} \left|\frac{\partial^2 \mathcal{P}(x)}{\partial x^2}\right|_{L_2(T_1^l - T_0^l)^\eta}.
\end{aligned}$$

It can be shown that Γ' is the dominating term and, by direct verification, we have $\Gamma'(T_1^l - T_0^l) = \mathbf{O}(\epsilon_2^{1-6\eta})$ as $T_1^l - T_0^l \to 0$, which proves our lemma. □

A. Finite Termination of Algorithm 3

We are now ready to show the finite termination of the line search Algorithm 3. If Algorithm 3 does not terminate, it must make infinitely many calls to the subdivision function. Otherwise, suppose only finitely many calls to the subdivision is used, Algorithm 3 reduces to a standard line search for NLP after the last call, which is guaranteed to succeed. However, we show that using infinitely many subdivisions will contradict the finiteness of $\mathcal{E}(\theta)$.

Lemma IX.4. *Taking Assumption III.1, IV.1, IV.3, if Algorithm 3 makes infinitely many calls to subdivision, then θ is unsafe. In other words, θ cannot pass the safety check Algorithm 2 under any finite spatial-temporal subdivision.*

Proof. Suppose θ is safe and Algorithm 3 is trying to update θ to $\theta' = \theta + d\alpha$. Such update must fail because only finitely many subdivisions are needed otherwise. Further, there must be some interval $[T_0^l, T_1^l]$ that requires infinitely many subdivisions. As a result, given any fixed ϵ_3 and ϵ_4, there must be some unsafe interval $[\bar{T}_0^l, \bar{T}_1^l] \subset [T_0^l, T_1^l]$ such that:

$$\bar{T}_1^l - \bar{T}_0^l \le \epsilon_3 \quad \alpha \le \epsilon_4.$$

We use the following shorthand notation:

$$\begin{aligned}
\bar{\mathcal{P}}_{ijkl} \triangleq & \mathcal{P}\left(\text{dist}\left(b_{ij}\left(\frac{\bar{T}_0^l + \bar{T}_1^l}{2}, \theta\right), o_k\right) - d_0\right) \\
\mathcal{P}'_{ijkl} \triangleq & \mathcal{P}\left(\text{dist}\left(b_{ij}\left(\frac{\bar{T}_0^l + \bar{T}_1^l}{2}, \theta'\right), o_k\right) - d_0\right)
\end{aligned}$$

$$\begin{aligned}
d_m \triangleq & \text{dist}\left(b_{ij}\left(\frac{\bar{T}_0^l + \bar{T}_1^l}{2}, \theta\right), o_k\right) - d_0 \\
d'_m \triangleq & \text{dist}\left(b_{ij}\left(\frac{\bar{T}_0^l + \bar{T}_1^l}{2}, \theta'\right), o_k\right) - d_0.
\end{aligned}$$

Since the interval is unsafe, we have:

$$\mathcal{P}'_{ijkl} \ge \mathcal{P}(\psi(\bar{T}_1^l - \bar{T}_0^l)).$$

Finally, we can bound the difference between penalty functions evaluated at θ and that at θ' using mean value theorem:

$$\begin{aligned}
&\left|\mathcal{P}'_{ijkl} - \bar{\mathcal{P}}_{ijkl}\right| = \left| \int_{d_m}^{d'_m} \frac{d\mathcal{P}(x)}{dx} dx \right| \\
\le & \max_{x \in [d_m, d'_m]} \left|\frac{d\mathcal{P}(x)}{dx}\right| |d_m - d'_m| \\
\le & L_4 \max_{x \in [d_m - L_4\|d\|\epsilon_4, d_m + L_4\|d\|\epsilon_4]} \left|\frac{d\mathcal{P}(x)}{dx}\right| \|d\| \epsilon_4.
\end{aligned}$$

The above result implies that the difference between the two penalty functions is controllable via ϵ_4. Since both ϵ_3 and ϵ_4 are arbitrary and independent, we can first choose small ϵ_4 such that:

$$\bar{\mathcal{P}}_{ijkl} = \mathbf{\Theta}(\mathcal{P}'_{ijkl}) \ge \mathbf{\Theta}(\mathcal{P}(\psi(\bar{T}_1^l - \bar{T}_0^l))) \ge \mathbf{\Theta}(\mathcal{P}(\psi(\epsilon_3))),$$

and then choose small ϵ_3 to make $\bar{\mathcal{P}}_{ijkl}$ arbitrarily large by Lemma IV.4. But for θ to be safe, we need:

$$\bar{\mathcal{P}}_{ijkl} \le \mathcal{P}(L_2 (T_1^l - T_0^l)^\eta),$$

leading to a contradiction, so θ cannot be safe. □

Corollary IX.5. *Algorithm 3 makes finitely many calls to subdivision, i.e., terminates finitely.*

Proof. Algorithm 1 requires the initial θ to be feasible, so the initial θ is safe. Each iteration of Algorithm 1 generates feasible iterations by Theorem V.1. If infinitely many subdivisions are used, then Lemma IX.4 implies that some θ is unsafe, which is a contradiction. □

B. Finite Termination of Algorithm 1

After showing the finite termination of Lemma IX.4, we move on to show the finite termination of main Algorithm 1. Our main idea is to compare $\tilde{\mathcal{P}}$ and $\hat{\mathcal{P}}$ and bound their difference. We first show that: $\hat{\mathcal{P}}$ is unbounded if infinite number of subdivisions are needed:

Lemma IX.6. *Taking Assumption III.1, IV.1, IV.3, for any fixed ϵ_5, if Algorithm 3 makes infinitely many calls to subdivision, then either $\hat{\mathcal{P}}(\epsilon_5)$ is unbounded or θ is unsafe.*

Proof. Following the same argument as Lemma IX.4, there must be unsafe interval $[\bar{T}_0^l, \bar{T}_1^l] \subset [T_0^l, T_1^l]$ with $\mathcal{P}_{ijkl} = \mathbf{\Theta}(\mathcal{P}'_{ijkl}) \ge \mathbf{\Theta}(\mathcal{P}(\psi(\epsilon_3))$ for any fixed ϵ_3. Since the domain is compact, there must be some $t \in [T_0^l, T_1^l]$ such that $\mathcal{P}_{ijk}(t) = \infty$. There are two cases for the interval $[T_0^l, T_1^l]$: (Case I) If $T_1^l - T_0^l < \epsilon_5$, then $\hat{\mathcal{P}}(\epsilon_5)$ is using the integral formula for the interval and $\mathcal{P}$ is unbounded by Lemma IV.4. (Case II) If $T_1^l - T_0^l \ge \epsilon_5$, then θ is unsafe by Lemma IX.4. □

Our final proof uses the Wolfe's condition to derive a contradiction if infinite number of subdivisions are needed. Specifically, we will show that the search direction is descendent if $\tilde{P}$ is replaced by $\hat{P}$ for some small ϵ_2. The following argument assumes $d = d^{(1)}$ and the case with $d = d^{(2)}$ follows an almost identical argument.

Proof of Theorem V.2. Suppose otherwise, we have $\|d\|_\infty \geq \epsilon_d$ because the algorithm terminates immediately otherwise. Due to the equivalence of metrics, we have $\|d\| \geq \epsilon_6$ for some ϵ_6. We introduce the following shorthand notation:

$$\hat{\mathcal{E}}(\theta, \epsilon_2) = \mathcal{O}(\theta) + \mu\hat{\mathcal{P}}(\epsilon_2).$$

We consider an iteration of Algorithm 1 that updates from θ to θ'. Since the first Wolfe's condition holds, we have:

$$\mathcal{E}(\theta') \leq \mathcal{E}(\theta) - c\|\nabla_\theta \mathcal{E}(\theta)\|^2\alpha \leq \mathcal{E}(\theta) - c\|\nabla_\theta \mathcal{E}(\theta)\|\alpha\epsilon_6.$$

The corresponding change in $\hat{\mathcal{E}}(\theta, \epsilon_2)$ can be bounded as follows:

$$\begin{aligned}
&\hat{\mathcal{E}}(\theta', \epsilon_2) = \hat{\mathcal{E}}(\theta, \epsilon_2) + \int_\theta^{\theta'} \left\langle \nabla_\theta \hat{\mathcal{E}}(\theta, \epsilon_2), d\theta \right\rangle \\
\leq &\hat{\mathcal{E}}(\theta, \epsilon_2) + \int_\theta^{\theta'} \left\langle \nabla_\theta \hat{\mathcal{E}}(\theta, \epsilon_2) - \nabla_\theta \mathcal{E}(\theta) + \nabla_\theta \mathcal{E}(\theta), d\theta \right\rangle \\
\leq &\hat{\mathcal{E}}(\theta, \epsilon_2) + \int_\theta^{\theta'} \|\nabla_\theta \hat{\mathcal{E}}(\theta, \epsilon_2) - \nabla_\theta \mathcal{E}(\theta)\| \|d\theta\| + \mathcal{E}(\theta') - \mathcal{E}(\theta) \\
\leq &\hat{\mathcal{E}}(\theta, \epsilon_2) + \mathbf{O}(\epsilon_2^{1-6\eta})\|d^{(1)}\|\alpha - c\|\nabla_\theta \mathcal{E}(\theta)\|\alpha\epsilon_6 \\
= &\hat{\mathcal{E}}(\theta, \epsilon_2) + \|\nabla_\theta \mathcal{E}(\theta)\|\alpha(\mathbf{O}(\epsilon_2^{1-6\eta}) - c\epsilon_6).
\end{aligned}$$

As long as $\eta < 1/6$, we can choose sufficiently small ϵ_2 such that $\hat{\mathcal{E}}(\theta', \epsilon_2) < \hat{\mathcal{E}}(\theta, \epsilon_2)$. Since we assume there are infinitely many subdivisions and Corollary IX.5 shows that line search Algorithm 3 will always terminate finitely, we conclude that Algorithm 1 will generate an infinite sequence θ of decreasing $\hat{\mathcal{E}}(\theta, \epsilon_2)$ for sufficiently small ϵ_2. Further, each θ is safe and each $\hat{\mathcal{E}}(\theta, \epsilon_2)$ is finite by the motion bound. But these properties contradict Lemma IX.6. □

X. Using Algorithm 1 as SIP Solver

We show that Algorithm 1 is indeed a solver of the SIP problem Equation (1). To this end, we consider running Algorithm 1 for an infinite number of iterations and we use superscript to denote iteration number. At the kth iteration, we use $\mu = \mu^k, \epsilon_d = \epsilon_d^k$ and we assume the sequences $\{\mu^k\}$ and $\{\epsilon_d^k\}$ are both null sequences. This will generate a sequence of solutions $\{\theta^k\}$ and we consider one of its convergent subsequence also denoted as $\{\theta^k\} \to \theta^0$. We consider the first-order optimality condition at θ^0:

Definition X.1. *If θ^0 satifies the first-order optimality condition, then for each direction D_θ such that:*

$$\left\langle D_\theta, \nabla_\theta \mathit{dist}(b_{ij}(t, \theta^0), o_k) \right\rangle \geq 0 \quad \forall \mathit{dist}(b_{ij}(t, \theta^0), o_k) = 0,$$

we have $\left\langle D_\theta, \nabla_\theta \mathcal{O}(\theta^0) \right\rangle \geq 0$.

We further assume the following generalized Mangasarian-Fromovitz constraint qualification (GMFCQ) holds at θ^0:

Assumption X.2. *There exists some direction D_θ and positive ϵ_7 such that:*

$$\left\langle D_\theta, \nabla_\theta \mathit{dist}(b_{ij}(t, \theta^0), o_k) \right\rangle \geq \epsilon_7 \quad \forall \mathit{dist}(b_{ij}(t, \theta^0), o_k) = 0.$$

MFCQ is a standard assumption establishing connection between the first-order optimality condition of NLP and the gradient of the Lagrangian function. Our generalized version of MFCQ requires a positive constant ϵ_7, which is essential for extending it to SIP. Note that GMFCQ is equivalent to standard MFCQ for NLP. We start by showing a standard consequence of assuming GMFCQ:

Lemma X.3. *Taking Assumption X.2, if first-order optimality fails at a trajectory θ^0, then we have a direction D_θ such that:*

$$\begin{aligned}
&\left\langle D_\theta, \nabla_\theta \mathcal{O}(\theta^0) \right\rangle < -\epsilon_8 \\
&\left\langle D_\theta, \nabla_\theta \mathit{dist}(b_{ij}(t, \theta^0), o_k) \right\rangle > \epsilon_9 \quad \forall \mathit{dist}(b_{ij}(t, \theta^0), o_k) = 0.
\end{aligned}$$

Proof. Under our assumptions, there is a direction D_θ^1 satisfying GMFCQ and another direction D_θ^2 violating first-order optimality condition. We can then consider a third direction $D_\theta^3 = D_\theta^1\epsilon_{10} + D_\theta^2$ where we have:

$$\begin{aligned}
&\left\langle D_\theta^3, \nabla_\theta \mathcal{O}(\theta^0) \right\rangle = \epsilon_{10} \left\langle D_\theta^1, \nabla_\theta \mathcal{O}(\theta^0) \right\rangle - \epsilon_{11} \\
&\left\langle D_\theta^3, \nabla_\theta \mathrm{dist}(b_{ij}(t, \theta^0), o_k) \right\rangle \geq \epsilon_{10}\epsilon_7,
\end{aligned}$$

where ϵ_{11} is some positive constant. We can thus choose sufficiently small ϵ_{10} to make the righthand side of the first equation negative and the righthand side of the second one positive. □

Assuming a failure direction D_θ^3 exists, we now start to show that the directional derivative of our objective function $\mathcal{E}$ along D_θ^3 is bounded away from zero, which contradicts the fact that our gradient norm threshold $\{\epsilon_d^k\}$ is tending to zero. To bound the derivative near θ^0, we need to classify $b_{ij}(t, \theta^0)$ into two categories: 1) its distance to o is bounded away from zero; 2) its distance to o is close to zero but b_{ij} is moving away along D_θ^3. This result is formalized below:

Lemma X.4. *Taking Assumption III.1, for D_θ^3 stated in Lemma X.3 and each tuple of $\langle i, j, k \rangle$, one of the following conditions holds:*

$$\begin{aligned}
&\mathit{dist}(b_{ij}(t, \theta^0), o_k) > \epsilon_{12} \\
&\mathit{dist}(b_{ij}(t, \theta^0), o_k) \leq \epsilon_{12} \wedge \left\langle D_\theta^3, \nabla_\theta \mathit{dist}(b_{ij}(t, \theta^0), o_k) \right\rangle \geq \frac{\epsilon_9}{2},
\end{aligned}$$

where ϵ_{12} is some positive constant.

Proof. Suppose otherwise, for arbitrarily small ϵ_{12}, we can find some i, j, k, t such that:

$$\begin{aligned}
&\mathrm{dist}(b_{ij}(t, \theta^0), o_k) \leq \epsilon_{12} \wedge \\
&\left\langle D_\theta^3, \nabla_\theta \mathrm{dist}(b_{ij}(t, \theta^0), o_k) \right\rangle < \frac{\epsilon_9}{2}.
\end{aligned} \tag{13}$$

We can construct a sequence of $\{\langle i, j, k, t, \epsilon_{12} \rangle\}$ with diminishing ϵ_{12} such that Equation (13) holds for each $\langle i, j, k, t, \epsilon_{12} \rangle$ tuple. If the sequence is finite, then there must be some:

$$\mathrm{dist}(b_{ij}(t, \theta^0), o_k) = 0 \wedge \left\langle D_\theta^3, \nabla_\theta \mathrm{dist}(b_{ij}(t, \theta^0), o_k) \right\rangle < \frac{\epsilon_9}{2},$$

contradicting Lemma X.3. If the sequence is infinite, then by Assumption III.1, there is an infinite subsequence with $\langle i,j,k\rangle$ being the same throughout the subsequence. We denote the subsequence as: $\{\langle t,\epsilon_{12}\rangle\}$, which tends to $\{\langle t^0,0\rangle\}$. By the continuity of functions we have:

$$\text{dist}(b_{ij}(t^0,\theta^0),o_k)=0\wedge\left\langle D^3_\theta,\nabla_\theta\text{dist}(b_{ij}(t^0,\theta^0),o_k)\right\rangle\leq\frac{\epsilon_9}{2},$$

again contradicting Lemma X.3. □

The above analysis is performed at θ^0. But by the continuity of problem data, we can extend the conditions to a small vicinity around θ^0. We denote $\mathcal{B}(\theta^0,\epsilon_{13})$ as a closed ball around θ^0 with a radius equal to ϵ_{13}. We formalize this observation in the following lemma:

Lemma X.5. *Taking Assumption X.2, III.1, and for D^3_θ stated in Lemma X.3, we have:*

$$\left\langle D^3_\theta,\nabla_\theta\mathcal{O}(\theta)\right\rangle<-\frac{\epsilon_8}{2},$$

and one of the following condition holds each tuple of $\langle i,j,k\rangle$:

$$dist(b_{ij}(t,\theta),o_k)>\frac{\epsilon_{12}}{2} \tag{14}$$

$$\left\langle D^3_\theta,\nabla_\theta dist(b_{ij}(t,\theta),o_k)\right\rangle\geq\frac{\epsilon_9}{4}, \tag{15}$$

for any $\theta\in\mathcal{B}(\theta^0,\epsilon_{13})$.

Proof. Combining Lemma X.3, X.4, and the continuity of problem data. □

Lemma X.5 allows us to quantify the gradient norm of $\mathcal{E}(\theta,\mu)$ (we write μ as an additional parameter of $\mathcal{E}$ for convenience). The gradient norm should tend to zero as $k\to\infty$. However, Lemma X.5 would bound it away from zero, leading to a contradiction.

Lemma X.6. *Taking Assumption X.2, III.1, IV.1, suppose $\{\mu^k\}$ is a null sequence, and for D^3_θ stated in Lemma X.3, there exists a D^3_θ and sufficiently large k such that for any $\theta\in\mathcal{B}(\theta^0,\epsilon_{14})$:*

$$\left\langle D^3_\theta,\nabla_\theta\mathcal{E}(\theta,\mu^k)\right\rangle<-\epsilon_{15},$$

for some positive constant ϵ_{14}.

Proof. The gradient consists of three sub-terms:

$$\nabla_\theta\mathcal{E}(\theta,\mu^k)=\nabla_\theta\mathcal{O}(\theta)+\nabla_\theta\bar{\mathcal{P}}_1(\theta,\mu^k)+\nabla_\theta\bar{\mathcal{P}}_2(\theta,\mu^k)$$

$$\bar{\mathcal{P}}_1(\theta,\mu^k)\triangleq\mu^k\sum_{ijkl}^{I}(T_1^l-T_0^l)\mathcal{P}_{ijkl}$$

$$\bar{\mathcal{P}}_2(\theta,\mu^k)\triangleq\mu^k\sum_{ijkl}^{II}(T_1^l-T_0^l)\mathcal{P}_{ijkl}.$$

Here we use $\sum_{ijkl}^{I}$ to denote a summation over terms $\mathcal{P}_{ijkl}$ where Equation (14) holds and $\sum_{ijkl}^{II}$ denotes a summation where Equation (14) does ont hold but Equation (15) holds. From Lemma X.5, we have the following holds for the first two terms:

$$\left\langle D^3_\theta,\nabla_\theta\mathcal{O}(\theta)\right\rangle<-\frac{\epsilon_8}{2}\quad\left\langle D^3_\theta,\nabla_\theta\bar{\mathcal{P}}_2(\theta,\mu^k)\right\rangle<0.$$

The third term can be arbitrarily small for sufficiently large k because:

$$\begin{aligned}&\left|\left\langle D^3_\theta,\nabla_\theta\bar{\mathcal{P}}_1(\theta,\mu^k)\right\rangle\right|\\ \leq&\mu^k\sum_{ijkl}^{I}(T_1^l-T_0^l)\left|\left\langle D^3_\theta,\left.\frac{\partial\mathcal{P}(x)}{\partial x}\right|_{\epsilon_{12}/2}\nabla_\theta\bar{\text{dist}}\right\rangle\right|\\ \leq&\mu^kT\left|\left.\frac{\partial\mathcal{P}(x)}{\partial x}\right|_{\epsilon_{12}/2}\right|\left|\left\langle D^3_\theta,\nabla_\theta\bar{\text{dist}}\right\rangle\right|.\end{aligned}$$

Here we have used Assumption IV.1 and denote $\nabla_\theta\bar{\text{dist}}$ as the location on trajectory where $\left|\left\langle D^3_\theta,\nabla_\theta\bar{\text{dist}}\right\rangle\right|$ takes the maximum value. Since $\{\mu^k\}$ is a null sequence, the third term can be arbitrarily small and the lemma follows. □

We are now ready to prove our main result:

Proof of Theorem V.3. Suppose otherwise, i.e. the first-order optimality condition fails, then the condition of Lemma X.6 holds, indicating that $\|\nabla_\theta\mathcal{E}(\theta^k,\mu^k)\|_\infty$ is bounded away from zero. But this contradicts the fact that $\{\epsilon_d^k\}$ is null. □